AD-754 543

GENERAL PURPOSE SIX DEGREE OF FREEDOM TERMINAL HOMING MISSILE SIMULATION PROGRAM

Lewis G. Minor

Army Missile Command
Redstone Arsenal, Alabama

31 August 1972

Best Available Copy

AD

AD 754543

TECHNICAL REPORT
RE-72-16

GENERAL PURPOSE SIX DEGREE OF FREEDOM TERMINAL HOMING MISSILE SIMULATION PROGRAM

by

Lewis G. Minor

31 August 1972

Approved for public release; distribution unlimited.

U.S. ARMY MISSILE COMMAND
Redstone Arsenal, Alabama

91

DISPOSITION INSTRUCTIONS

Destroy this report when it is no longer needed. Do not return it to the originator.

DISCLAIMER

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

TRADE NAMES

Use of trade names or manufacturers in this report does not constitute an official indorsement or approval of the use of such commercial hardware or software.

UNCLASSIFIED
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Terminal Homing Missile Simulation						
Six Degree of Freedom						
Quaternions						
Digital program						
FORTRAN						

16

UNCLASSIFIED
Security Classification

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Advanced Sensors Directorate Directorate for Research, Development Engineering and Missile System Laboratory U.S. Army Missile Command Redstone Arsenal, Alabama 35809	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP N/A

3. REPORT TITLE

GENERAL PURPOSE SIX DEGREE OF FREEDOM TERMINAL HOMING MISSILE SIMULATION PROGRAM

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Technical Report

5. AUTHOR(S) (First name, middle initial, last name)

Lewis G. Minor

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
31 August 1972	92	4

8a. CONTRACT OR GRANT NO. b. PROJECT NO. (DA) 1M262303A214 AMC Management Structure Code c. 632303.11.21403 d.	9a. ORIGINATOR'S REPORT NUMBER(S) RE-72-16 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) AD________

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
None	Same as No. 1

13. ABSTRACT

This report contains a description of a general purpose, 6-degree of freedom terminal homing missile simulation. The program uses quaternions for the coordinate transformations and features the use of subroutines to enter seeker, autopilot, aerodynamic, and wind models. The program is written in FORTRAN.

DD FORM 1 NOV 65 1473 REPLACES DD FORM 1473, 1 JAN 64, WHICH IS OBSOLETE FOR ARMY USE.

UNCLASSIFIED

Security Classification

ia

31 August 1972

Technical Report RE-72-16

GENERAL PURPOSE SIX DEGREE OF FREEDOM TERMINAL HOMING MISSILE SIMULATION PROGRAM

by

Lewis G. Minor

DA Project No. 1M262303A214

AMC Management Structure Code 632303.11.21403

Approved for public release; distribution unlimited.

Advanced Sensors Directorate

Directorate for Research, Development

Engineering and Missile Systems Laboratory

U.S. Army Missile Command,

Redstone Arsenal, Alabama 35809

ABSTRACT

This report contains a description of a general purpose, 6-degree of freedom terminal homing missile simulation. The program uses quaternions for the coordinate transformations and features the use of subroutines to enter seeker, autopilot, aerodynamic, and wind models. The program is written in FORTRAN.

CONTENTS

Section I INTRODUCTION

1. Purpose of Program

The General Purpose, 6-Degree of Freedom Terminal Homing Missile Simulation Program is a FORTRAN program designed to simulate the dynamics of terminal homing missile systems with a maximum degree of flexibility. The flexibility is achieved through the use of user supplied subprograms which are used in conjunction with the main program.

The program uses quaternions, as opposed to Euler angle rotations, to generate the coordinate system transformations. The quaternion approach is as accurate as the Euler angle approach[1] and has the advantage of avoiding "gimbal lock" which may be encountered in some cases with Euler angle rotations. The quaternions do not need FORTRAN SIN or COS routines in the computation of the transformations matrices which results in a potential savings in the computation time.

2. Assumptions

The airframe is assumed to be a rigid body and aeroelastic effects are not included.

The airframe is assumed to have a plane of mass symmetry coinciding with the vertical plane of reference (plane defined by the missile x and z axes in Figure 1). The y-axis is therefore a principal axis and the products of inertial I_{xy} and I_{yz} vanish. Thus, if mass asymmetries are to be simulated, the xy-plane must be used (xz plane must be a plane of symmetry).

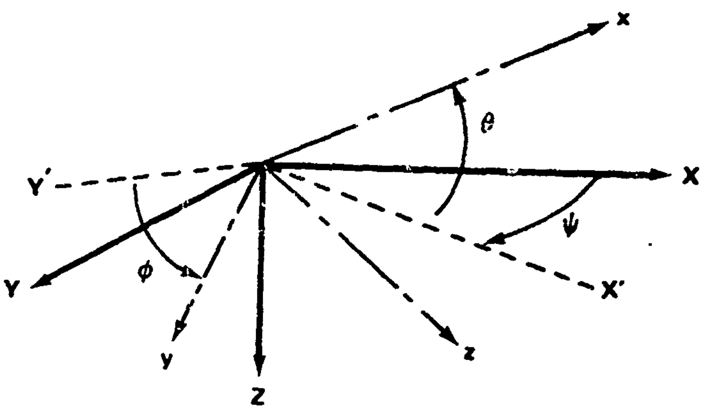
A flat nonrotating earth with constant gravity is assumed.

3. Coordinate Systems Used by the Program

The program uses two coordinate systems. The earth fixed (inertial system) and the missile body-axis system. The earth fixed coordinate system is assumed to be fixed to a flat earth with the z-axis of a right hand coordinate system pointing down. The missile body-axis system is a right hand coordinate system with the x-axis aligned with the longitudinal axis of the missile. The coordinate system is fixed to the missile and rolls with it. The relationship between the earth

---

[1] Robinson, A. C., On the Use of Quaternions in Simulation of Rigid Body Motion, WADC TR 58-17, December 1958.



X,Y,Z - EARTH SYSTEM AXES
x,y,z - BODY AXIS SYSTEM AXES
θ,ϕ,ψ - EULER ANGLES
X',Y',Z' - INTERMEDIATE AXES IN ROTATIONAL SEQUENCE

Figure 1. Relationship Between the Earth and Body-Axis Systems

fixed and body-axis systems is given in Figure 1 in terms of Euler angles. However, Euler angles are not used in the computational structure of the program except for the initial conditions and when Euler angle printout is selected as part of a standard printout option (Section V).

4. Six Degrees of Freedom in the Simulation

The 6 degrees of freedom of the airframe consist of three translations along the inertial X, Y, and Z axes (earth fixed system) and three rotations about the body-fixed x, y, and z axes. The rotations are expressed by the quaternion $e_0 + ie_1 + je_2 + ke_3$. (Appendix A contains a discussion of quaternions.)

Section II EQUATIONS USED BY THE PROGRAM

1. Variable List

The following is a list of variables which are used in the equations in the simulation along with their corresponding FORTRAN program variable name.

EQUATION VARIABLE	FORTRAN VARIABLE	VARIABLE DEFINITION
a_1	A(1)	Element of the coordinate transformation matrix
a_2	A(2)	Element of the coordinate transformation matrix
a_3	A(3)	Element of the coordinate transformation matrix
α	ALFA	Angle of attack in the pitch plane
b_1	A(4)	Element of the coordinate transformation matrix
b_2	A(5)	Element of the coordinate transformation matrix
b_3	A(6)	Element of the coordinate transformation matrix
β	BETA	Angle of attack in the yaw plane
c_1	A(7)	Element of the coordinate transformation matrix
c_2	A(8)	Element of the coordinate transformation matrix
c_3	A(9)	Element of the coordinate transformation matrix
δ_1	DP1	Wing deflection fin 1 (pitch)
δ_2	DY2	Wing deflection fin 2 (yaw)
δ_3	DP3	Wing deflection fin 3 (pitch)
δ_4	DY4	Wing deflection fin 4 (yaw)
D	D	Diameter of missile body
ΔX	DELXE	Missile to target displacement in earth X direction
ΔY	DELYE	Missile to target displacement in earth Y direction

ΔZ	DELZE	Missile to target displacement in earth Z direction
Δt	DT	Integration step size
Δt_{min}	DTMIN	Minimum allowed integration step size
E_{max}	EMAX	Maximum allowed error in integration
e_0	X(4), EO	Quaternion parameter
e_1	X(5), E1	Quaternion parameter
e_2	X(6), E2	Quaternion parameter
e_3	X(7), E3	Quaternion parameter
ϵ	-	Constraint error [See Equation (II-12)]
f_x	FX	Body-axis component of force in x direction
f_y	FY	Body-axis component of force in y direction
F_y	FYE	Earth system component of force in Y direction
f_z	FZ	Body-axis component of force in z direction
F_z	FZE	Earth system component of force in Z direction
g	G	Acceleration due to gravity
h_x	HX	Angular momentum of missile about body-axis x-axis
h_y	HY	Angular momentum of missile about body-axis y-axis
h_z	HZ	Angular momentum of missile about body-axis z-axis
-	ITERA	Number of integration iterations
I_x	IX	Missile's moment of inertia about body-axis x-axis
I_y	IY	Missile's moment of inertia about body-axis y-axis
I_z	IZ	Missile's moment of inertia about body-axis z-axis

I_{zx}	IZX	Missile's product of inertia in the x-z plane
K	K	Arbitrary constant used in quaternion constraint
-	KE	KE = (K) (ϵ)
m	MASS	Mass of the missile
M	MACH	Mach number of the missile
M_x	MX	Moment about the body-axis x-axis
M_y	MY	Moment about the body-axis y-axis
-	MAXPT	Maximum allowed number of printouts
M_z	MZ	Moment about the body-axis z-axis
n_s	SX	Number of seeker and autopilot state variables
n_t	TX	Number of target state variables
p	X(1), P	Angular rate about the body-axis x-axis
-	PRNTI	Print interval
ϕ	PHI	Euler angle rotation phi
ψ	PSI	Euler angle rotation psi
qs, qsd	QS, QSR, QSD	Dynamic pressure terms
q	X(2), Q	Angular rate about the body-axis y-axis
ρ	RHO	Air density constant
r	X(3), R	Angular rate about the body-axis z-axis
R_{min}	RMIN	Miss distance (computer after run is complete)
-	R(4)	Integration status parameter (See Appendix B)
-	RM	Range to the target
S	S	Reference area of missile
σ_y	SIGY	Line of sight angle in earth coordinate system X-Z plane

σ_z	SIGZ	Line of sight angle in earth coordinate system X-Y plane
-	SMAX	Maximum error in integration computation
S_i	X(I), $14 \le I \le n_s + 13$	Seeker and autopilot state variables
T_i	X(I), $n_s + 14 \le I$ and $I \le n_s + n_t + 13$	Target state variables
-	TMAX	Maximum time for a computer run (real time)
θ	THTA	Euler rotation Theta
u	U	Body-axis component of missile velocity in the x direction
U	X(8)	Earth system component of missile velocity in the X direction
U_e	UE	Body system component of missile airspeed in the x direction
v	V	Body-axis component of missile velocity in the y direction
V	X(9)	Earth system component of missile velocity in the Y direction
V_e	VE	Earth system component of missile airspeed in the Y direction
V_m	VM	Velocity of the missile
V_s	VS	Velocity of sound
w	W	Body-axis component of missile in the z direction
W	X(10)	Earth system component of missile velocity in the Z direction
W_e	WE	Earth system component of missile airspeed in Z direction

W_x	WX	Earth system component of wind velocity in X direction
W_y	WY	Earth system component of wind velocity in Y direction
W_z	WZ	Earth system component of wind velocity in Z direction
x	X	Body-axis component of missile displacement in the x direction
X	X(11), XE	Earth system component of missile displacement in the X direction
X_T	XTE	Earth system component of missile target in the X direction
y	Y	Body-axis component of missile displacement in y direction
Y	X(12), YE	Earth system component of missile displacement in Y direction
Y_T	YTE	Earth system component of target displacement in Y direction
z	Z	Body-axis component of missile displacement in z direction
Z	X(13), ZE	Earth system component of missile displacement in Z direction
Z_T	ZTE	Earth system component of target displacement in Z direction

A "dot" over a variable will be used to indicate the time derivative of the variable. For example $\dot{X}$ is the time rate change of X. The derivative of the FORTRAN variable X(I) will be indicated by the variable DX(I).

2. Initial Conditions

The initial conditions for the simulation are given in terms of the equation variables: X, Y, Z, u, v, w, θ, ϕ, ψ, p, q, r, S_1,

$S_2, \ldots S_{n_s}$. $T_1, T_2, \ldots T_{n_t}$ which are supplied by the user. The equation variables to be integrated are:

$$\dot{X}, \dot{Y}, \dot{Z}, \ddot{X}, \ddot{Y}, \ddot{Z}, \dot{p}, \dot{q}, \dot{r}, \dot{e}_0, \dot{e}_1, \dot{e}_2, \dot{e}_3, \dot{S}_1, \dot{S}_2, \ldots \dot{S}_{n_s}; \dot{T}_1, \dot{T}_2, \ldots \dot{T}_{n_t}.$$

Thus, $\dot{X}$, $\dot{Y}$, $\dot{Z}$, e_0, e_1, e_2 and e_3 must be computed from the input quantities supplied by the user, and Equation (A-33) of Appendix A, gives

$$e_0 = \cos\psi/2 \cos\theta/2 \cos\phi/2 + \sin\psi/2 \sin\theta/2 \sin\phi/2$$

$$e_1 = \cos\psi/2 \cos\theta/2 \sin\phi/2 - \sin\psi/2 \sin\theta/2 \cos\phi/2$$

$$e_2 = \cos\psi/2 \sin\theta/2 \cos\phi/2 + \sin\psi/2 \cos\theta/2 \sin\phi/2$$

$$e_3 = -\cos\psi/2 \sin\theta/2 \sin\phi/2 + \sin\psi/2 \cos\theta/2 \cos\phi/2 \quad . \text{(II-1)}$$

To find $\dot{X}$, $\dot{Y}$, $\dot{Z}$ we must generate a transformation matrix which will take the velocities u, v, and w (body-axis system) into $\dot{X}$, $\dot{Y}$, $\dot{Z}$ (inertial reference system). Thus, using Equation (A-29) of Appendix A,

$$a_1 = e_0^2 + e_1^2 - e_2^2 - e_3^2$$

$$a_2 = 2\left(e_1 e_2 + e_0 e_3\right)$$

$$a_3 = 2\left(e_1 e_3 - e_0 e_2\right)$$

$$b_1 = 2\left(e_1 e_2 - e_0 e_3\right)$$

$$b_2 = e_0^2 + e_2^2 - e_1^2 - e_3^2$$

$$b_3 = 2\left(e_2 e_3 + e_0 e_1\right)$$

$$c_1 = 2\left(e_1 e_3 + e_0 e_2\right)$$

$$c_2 = 2\left(e_2 e_3 - e_0 e_1\right)$$

$$c_3 = e_0^2 + e_3^2 - e_1^2 - e_2^2 \quad . \qquad \text{(II-2)}$$

Then

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \quad . \qquad \text{(II-3)}$$

3. Computational Sequence

a) Compute

$$\alpha = \tan^{-1}\left(\frac{w}{u}\right)$$

$$\beta = \tan^{-1}\left(\frac{v}{w}\right) \qquad \text{(II-4)}$$

b) Compute V_s and ρ as a function of Z by linear interpolation

c) Compute

$$M = \sqrt{\frac{u^2 + v^2 + w^2}{V_s}} \qquad \text{(II-5)}$$

d) Compute seeker, autopilot, and target dynamics from user supplied equations of the form

$$\begin{bmatrix} \dot{S}_1 \\ \dot{S}_2 \\ \vdots \\ \dot{S}_{n_s} \end{bmatrix} = \begin{bmatrix} g_1\left(S_1, S_2, \ldots, S_{n_s}\right) \\ g_2\left(S_1, S_2, \ldots, S_{n_s}\right) \\ g_{n_s}\left(S_1, S_2, \ldots, S_{n_s}\right) \end{bmatrix} \quad . \qquad \text{(II-6)}$$

$$\begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ \delta_4 \end{bmatrix} = \begin{bmatrix} h_1\left(S_1, S_2, \ldots, S_{n_s}\right) \\ h_2\left(S_1, S_2, \ldots, S_{n_s}\right) \\ h_3\left(S_1, S_2, \ldots S_{n_s}\right) \\ h_4\left(S_1, S_2, \ldots S_{n_s}\right) \end{bmatrix} \tag{II-7}$$

$$\begin{bmatrix} \dot{T}_1 \\ \dot{T}_2 \\ \dot{T}_{n_t} \end{bmatrix} = \begin{bmatrix} f_1\left(T_1, T_2, \ldots, T_{n_t}\right) \\ f_2\left(T_1, T_2, \ldots, T_{n_t}\right) \\ f_{n_t}\left(T_1, T_2, \ldots, T_{n_t}\right) \end{bmatrix} \tag{II-8}$$

$$\begin{bmatrix} X_T \\ Y_T \\ Z_T \end{bmatrix} = \begin{bmatrix} z_1\left(T_1, T_2, \ldots, T_{n_t}\right) \\ z_2\left(T_1, T_2, \ldots, T_{n_t}\right) \\ z_3\left(T_1, T_2, \ldots, T_{n_t}\right) \end{bmatrix} . \tag{II-9}$$

e) The forces and moments in the body axis system - f_x, f_y, f_z, M_x, M_y, and M_z - are computed by user supplied equations of α, β, S, D, M, δ_1, δ_2, δ_3, δ_4, and ρ. The moments of inertia I_x and I_y, the product of inertia I_{zx}, and the mass m of the missile are also supplied by user.

f) Compute the following equations for missile angular accelerations:

$$\dot{p} = \frac{\dot{h}_x I_z + \dot{h}_z I_{zx}}{I_x I_z - I_{zx}^2}$$

$$\dot{q} = \frac{\dot{h}_y}{I_y}$$

$$\dot{r} = \frac{\dot{h}_z\ I_x + \dot{h}_x\ I_{zx}}{I_x\ I_z - I_{zx}^2} \qquad \text{(II-10)}$$

where

$$\dot{h}_x = M_x - qr\ \left(I_z - I_y\right) + qp\ I_{zx}$$

$$\dot{h}_y = M_y - pr\ \left(I_x - I_z\right) - \left(p^2 - r^2\right) I_{zx}$$

$$\dot{h}_z = M_z - pq\ \left(I_y - I_x\right) - qr\ I_{zx} \qquad \text{(II-11)}$$

g) Compute quaternion derivatives, Equation (A-74) of Appendix A

$$\dot{e}_0 = -1/2\ \left(e_1\ p + e_2\ q + e_3\ r\right) + K\ \epsilon\ e_0$$

$$\dot{e}_1 = 1/2\ \left(e_0\ p + e_2\ r - e_3\ q\right) + K\ \epsilon\ e_1$$

$$\dot{e}_2 = 1/2\ \left(e_0\ q + e_3\ p - e_1\ r\right) + K\ \epsilon\ e_2$$

$$\dot{e}_3 = 1/2\ \left(e_0\ r + e_1\ q - e_2\ p\right) + K\ \epsilon\ e_3 \qquad \text{(II-12)}$$

where

$$\epsilon = 1 - \left(e_0^2 + e_1^2 + e_2^2 + e_3^2\right)$$

and where K is an arbitrary constant (K = 100 in this program).

h) Compute the forces in the inertial system

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix} \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \quad . \qquad \text{(II-13)}$$

i) Compute the accelerations in the inertial system

$$\dot{U} = F_x/m$$

$$\dot{X} = U$$

$$\dot{V} = F_y/m$$

$$\dot{Y} = F_z/m + g$$

$$\dot{Z} = W \quad . \qquad \text{(II-14)}$$

j) Integrate the following equations variables:

$$\dot{p},\ \dot{q},\ \dot{r},\ \dot{U},\ \dot{V},\ \dot{W},\ \dot{X},\ \dot{Y},\ \dot{Z},\ \dot{e}_0,\ \dot{e}_1,\ \dot{e}_2,\ \dot{e}_3 \quad .$$

k) Compute the components of wind speed W_x, W_y, W_z in the inertial system by user supplied equations.

l) Compute the inertial components of missile airspeed

$$U_e = U - W_x$$

$$V_e = V - W_y$$

$$W_e = W - W_z \quad . \qquad \text{(II-15)}$$

m) Compute the body axis components of missile airspeed (u, v, and w).

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix} \begin{bmatrix} U_e \\ V_e \\ W_e \end{bmatrix} \tag{II-16}$$

where

$$a_1 = e_0^2 + e_1^2 - e_2^2 - e_3^2$$

$$a_2 = 2\left(e_1e_2 + e_0e_3\right)$$

$$a_3 = 2\left(e_1e_3 - e_0e_2\right)$$

$$b_1 = 2\left(e_1e_2 - e_0e_3\right)$$

$$b_2 = e_0^2 + e_2^2 - e_1^2 - e_3^2$$

$$b_3 = 2\left(e_2e_3 + e_0e_1\right)$$

$$c_1 = 2\left(e_1e_3 + e_0e_2\right)$$

$$c_2 = 2\left(e_2e_3 - e_0e_1\right)$$

$$c_3 = e_0^2 + e_3^2 - e_1^2 - e_2^2 \quad . \tag{II-17}$$

m) The computational loop is closed by going back to Step a.

Section III FORTRAN SUBROUTINES SUPPLIED BY USER

1. General Information

The user must supply the following subroutines: SEEKER, TARGET, FOROM, WRT, STRPLT, and WIND. Some of these subroutines may not be required and may consist of only a DIMENSION, a RETURN, and an END statement.

Variables not in the argument list for these subroutines must be "transferred" by COMMON statements. The main program contains some standard common blocks which can be used as needed (Paragraph 2 of Section V).

2. Subroutine SEEKER

Subroutine SEEKER is used to model the dynamics of the seeker and autopilot sections. In most cases common block ANG will be required because the wing deflections will be needed in other subroutines and in the main program when the standard printout option is used. Section V of this report contains additional information on the standard common blocks.

Subroutine SEEKER should contain a model of the form given in step d) of Paragraph 3 of Section II. The subroutine should define the derivatives of the state variable to be integrated by using the FORTRAN variable DX(I) where I = 14, 15, . . ., SX + 13.

For example, let us assume that the seeker and autopilot section are defined by the block diagram given below (the s is a Laplace operator.

Let the variables shown in Figure 2 be defined as follows:

σ_y - Line of sight angle in inertial space for the X-Z plane (rad)

σ_z - Line of sight angle in inertial space for the Y-Z plane (rad)

δ_p - Pitch wing command (rad)

δ_y - Yaw wing command (rad)

K - Seeker gain (rad/sec/rad)

K_n - Navigation gain.

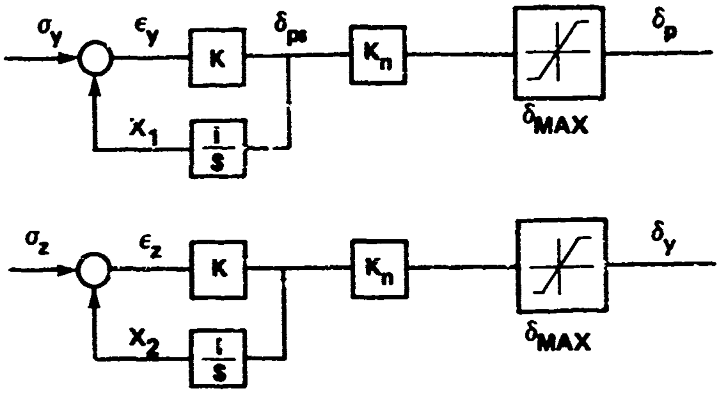


Figure 2. Simplified Block Diagram Example

The differential equations for the block diagram in Figure 2 can be written as a set of first order differential equations obtained directly from Figure 2, and

$$\dot{X}_1 = K\left(\sigma_y - X_1\right)$$

$$\delta_p = K_n \dot{X}_1 \quad \text{when} \quad |K_n \dot{X}_1| \le \delta_{max}$$

$$\delta_p = \left(\delta_{max}\right) \mathrm{SGN}\left(\dot{X}_1\right) \quad \text{when} \quad |K_n \dot{X}_1| > \delta_{max}$$

$$\dot{X}_2 = K\left(\sigma_z - X_2\right)$$

$$\delta_y = K_n \dot{X}_2 \quad \text{when} \quad |K_n \dot{X}_2| \le \delta_{max}$$

$$\delta_y = \left(\delta_{max}\right) \mathrm{SGN}\left(\dot{X}_2\right) \quad \text{when} \quad |K_n \dot{X}_2| > \delta_{max} \quad . \qquad \text{(III-1)}$$

Let $\delta_p = \delta_y = 0$ if time < 50 seconds.

The corresponding FORTRAN program is:

```
SUBROUTINE SEEKER(TIME,X,DX)
REAL INPUT
REAL KE
REAL K,KN
DIMENSION X(1),DX(1)
COMMON /ANG/ALFA,BETA,SIGY,SIGZ,DP1,DY2,DP3,DY4
KN=8.5
K=1Ø.
DX(14)=K*(SIGY-X(14))
DX(15)=K*(SIGZ-X(15))
DP1=KN*DX(14)
DY2=KN*DX(15)
IF(ABS(DP1).GT..1745)  DP1=SIGN(.1745,DP1)
IF(ABS(DY2).GT..1745)  DY2=SIGN(.1745,DY2)
IF(TIME.LT.5Ø.)  DP1=DY2=Ø.
DY4=DY2
DP3=DP1
RETURN
END
```

Note that the wing commands DP1, DP3, DY2, and DY4 may be defined in any manner the user wishes, however, the user must be consistent with the wing command defined in the user supplied FOROM subroutine. Note also that the FORTRAN variable SX is equal to two because there are two seeker states [X(14) and X(15)].

3. Subroutine TARGET

Subroutine TARGET is used to model the target dynamics. The model should have the form of the model given in step d), Paragraph 3 of Section II. The subroutine as a general rule should contain the standard common block DISPL (Section V). The derivatives of the state variables to be integrated must be defined by the FORTRAN DX(I), where I = 13 + SX + 1, 13 + SX + 2, ..., 13 + SX + TX. The target locations in earth coordinate system (XTE, YTE, and ZTE) must also be defined.

Subroutine TARGET can be illustrated by a simple example where the target is located at X = 46,000 ft, Y = 500 ft, Z = 0 ft in the earth coordinate system and is moving with a velocity of 5 ft/sec in the Y direction.

```
SUBROUTINE TARGET(TIME,X,DX)
DIMENSION X(1),DX(1)
COMMON /DISPL/ U,V,W,DELXE,DELYE,DELZE,XTE,YTE,ZTE,RM
DX(16)=5.
XTE=46000.
YTE=500.+X(16)
ZTE=0.
RETURN
END
```

Note that there is one state variable in the subroutine and the FORTRAN variable TX is equal to 1.

4. Subroutine FOROM

The purpose of subroutine FOROM is to supply the forces, moments, moments of inertia, product of inertia, and mass to the main program. The forces are in the body-axis system.

The positive conventions for forces, moments, and angular rates for the body-axis system are shown in Figure 3.

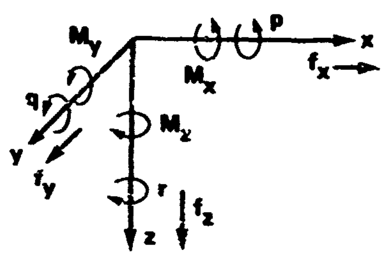


Figure 3. Positive Conventions in the Body-Axis System

Subroutine FOROM must contain common blocks: FRMO, AIR, ANG, and OTHER2 for most applications (Section V) likely to be encountered.

The use of subroutine FOROM can be illustrated by a simple example:

```
SUBROUTINE FOROM(TIME,X,DX)
INTEGER SX,TX
REAL KE,K
REAL MACH,MX,MY,MZ,IX,IY,IZ,IZX,MASS
DIMENSION X(1),DX(1)
COMMON /ANG/ALFA,BETA,SIGY,SIGZ,DP1,DY2,DP3,DY4
COMMON /OTHER2/ THTA,PSI,PHI,KE,S,D,SX,TX,K,G,MAXPT
COMMON /AIR/ MACH,VM,VS,QS,QSD,QSR,RHO
```

```
COMMON/FRMO/ FX,FY,FZ,MX,MY,MZ,IX,IY,IZ,IZX,MASS
IY=IZ=4.77
IX=.202
MASS=4.671
IZX=0.
S=.1965
D=.5
CD0=.25
CMA=-23.
CNA=15.
CMD=18.
CMQ=-283.
CALL QSDSUB
FX=-.5*QS*CDO
FY=-.5*QS*CNA*BETA
FZ=-.5*QS*CNA*ALFA
Q=X(2)
R=X(3)
MX=0.
MY=.5*QSD*(CMA*ALFA+CMD*DP1+D*CMQ*Q/(2.*VM))
MZ=.5*QSD*(-CMA*BETA+CMD*DY2+D*CMQ*R/(2.*VM))
RETURN
END
```

Note in the example the aerodynamic coefficients are assumed to be constants. In most simulations this assumption would not be valid and aero data must be stored in tables as a function of Mach number. For some tail controlled missiles where the angles of attack are likely to be relatively large, it may be necessary to compute the aerodynamic coefficients as a function of three independent variables - Mach number, angle of attack, and wing deflection. The problem is basically one of interpolating to find an accurate aerodynamic coefficient for any given set of independent variables. To help solve this problem, an interpolation routine has been provided as part of the main program which will handle one, two, or three independent variables. A description of the interpolation routine can be found in Section IV. Subroutine QSDSUB is a subroutine supplied by the main program which computes the dynamic pressure terms (Section IV).

5. Subroutine WRT

The purpose of subroutine WRT is to provide a way to output certain variables not in the standard printout. As will be discussed in Section V, there is an input data card which provides the user with the following output options: standard printout only, printout provided by the user in WRT only, or the printout provided by the standard printout plus the output generated by WRT. Subroutine WRT must as a minimum contain

```
SUBROUTINE WRT(TIME,X,DX)
DIMENSION X(1),DX(1)
RETURN
END
```

For this example, the standard printout option would normally be used. All variables to be printed out by WRT with the exception of the arrays X and DX, and the variable TIME must be transferred to WRT by COMMON statements.

6. Subroutine STRPLT

The purpose of the user supplied subroutine STRPLT is to store output data in arrays as required and to provide enough flexibility so that plots can be made with either the line printer or with some other output device such as the Tektronix 4002A graphics terminal.

A line printer plot will be generated by the following example program. Subroutine PLOT is discussed in Section IV. The line printer plot generated by this program is given in Section VI.

```
      SUBROUTINE STRPLT(TIME,X,DX,IMISS,IPLOT)
      DIMENSION X(1),DX(1)
      DIMENSION A(1000),B(500)
      COMMON/OTHER1/ ITERA,DT,DTMIN,EMAX,N,SMAX,TMAX,PRNTI
4     FORMAT(1H1,50X,*ZE VS TIME*// )
      IF(ITERA.NE.1)  GO TO 1
      I=0
      TS=0.
      STEP=1.
1     CONTINUE
      IF(TIME.LT.TS.AND.IPLOT.EQ.0)  GO TO 3
      TS=TIME+STEP
      I=I+1
      A(I)=TIME
      B(I)=X(13)
      IF(IPLOT.EQ.0)  GO TO 3
      WRITE(6,4)
      N=I
      DO 5 J=1,N
5     A(J+N)=B(J)
      CALL PLOT(A,N,2,2,3)
3     RETURN
      END
```

The FORTRAN variable IPLOT is set to zero during the flight phase of the simulation and is set to one when the run is complete and plotting can commence. IMISS = 0 when the miss distance is less than 50 feet. IMISS = 1 when the miss distance is greater than 50 feet (RMIN has no meaning).

7. Subroutine WIND

The purpose of the user supplied subroutine WIND is to provide a method for placing a wind model in the simulation when it is required.

Subroutine WIND should contain a minimum of the following

```
SUBROUTINE WIND(TIME,X,DX,U,V,W)
DIMENSION X(1),DX(1)
RETURN
END
```

The variable U, V, and W are components of missile velocity in the earth coordinate system. So to enter a wind model; U, V, and W must be "replaced" with inertial components of missile airspeed. For example, if the wind is blowing in the inertial Y direction with a velocity of 10 ft/sec, the subroutine would take the following form:

```
SUBROUTINE WIND (TIME,X,DX,U,V,W)
DIMENSION X(1),DX(1)
V=V-10.
RETURN
END
```

Section IV FORTRAN SUBROUTINES SUPPLIED BY PROGRAM

1. General Information

There are certain subroutines available with the main program which may be called by the user supplied subroutines. The subroutines simplify the task of programming in that a group of statements may be replaced by a single call statement in the user's program. The following sections define the function of each subroutine which can be called by the user.

2. Subroutine LIM

Subroutine LIM contains a limiter model and has a calling statement of the form

CALL LIM (INPUT, OUTPUT, A, B)

where

$$\begin{aligned} &\text{OUTPUT} = \text{INPUT} \quad \text{when } A \leq \text{INPUT} \leq B \\ &\text{OUTPUT} = B \quad \text{when INPUT} > B \\ &\text{OUTPUT} = A \quad \text{when INPUT} < A \end{aligned} \tag{IV-1}$$

NOTE: INPUT is a real variable in subroutine LIM.

3. Subroutine LIMSTA

Subroutine LIMSTA contains a limiter model which modifies the input quantity as well as limits the output. The calling statement is of the form

CALL LIMSTA (INPUT, OUTPUT, A, B)

where

$$\begin{aligned} &\text{OUTPUT} = \text{INPUT} \quad \text{when } A \leq \text{INPUT} \leq B \\ &\text{OUTPUT} = \text{INPUT} = B \quad \text{when INPUT} > B \\ &\text{OUTPUT} = \text{INPUT} = A \quad \text{when INPUT} < A \end{aligned} \tag{IV-2}$$

NOTE: INPUT is a real variable in subroutine LIMSTA.

4. Subroutine DETEC

Subroutine DETEC contains a simplified detector model with a characteristic as shown in Figure 4.

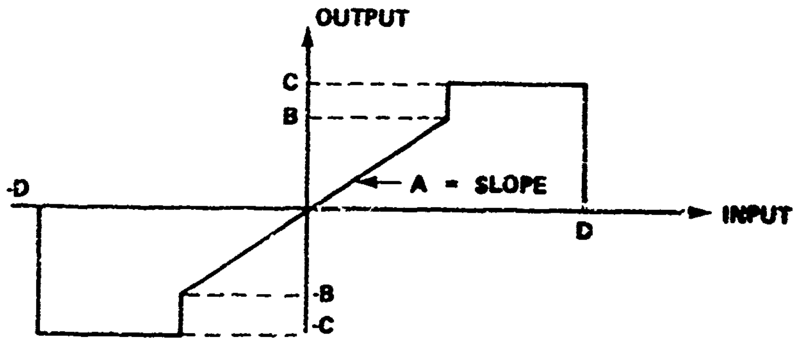


Figure 4. Model of Detector Characteristic

The calling statement is of the form

CALL DETEC (INPUT, OUTPUT, A, B, C, D)

where the arguments are defined in Figure 4.

NOTE: INPUT is a real variable in subroutine DETEC.

5. Subroutine DEADSP

Subroutine DEADSP contains a model of an element with dead space as shown in Figure 5. The calling statement is of the form

CALL DEADSP (INPUT, OUTPUT, A, B)

where the arguments are defined in Figure 5.

NOTE: INPUT is a real variable in subroutine DEADSP.

6. Subroutine LAG

Subroutine LAG contains a model of a first order lag with a transfer function of the form

$$\frac{\text{OUTPUT(S)}}{\text{INPUT(S)}} = \frac{1}{\tau s + 1} \quad . \qquad \text{(IV-3)}$$

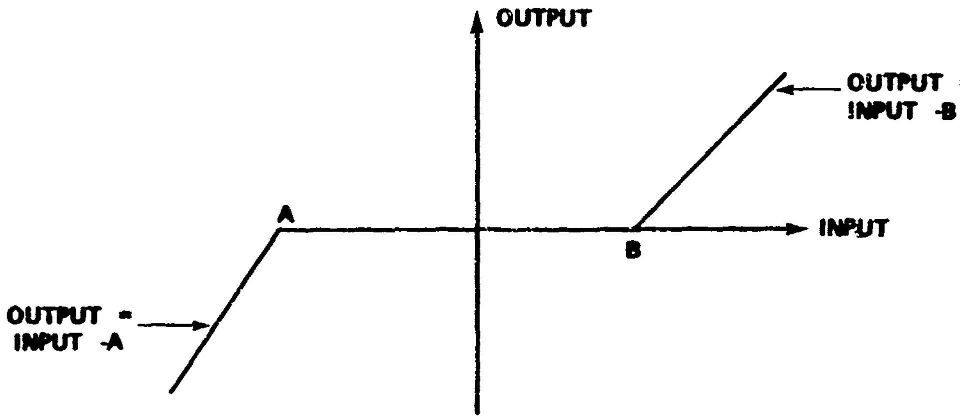


Figure 5. Dead Space Model

The calling statement is of the form

CALL LAG (INPUT, OUTPUT, X, DX, INDE T)

where X is the array of state variables used by the main program (similarly DX is the array consisting of the derivative of X), INDEX is an integer defining the element of X which defines the state of the first order lag, and T is the time constant τ defined in Equation (IV-3). Note that a first order lag model requires only one state variable and that INPUT is a real variable in subroutine LAG.

7. Subroutine SECORD

Subroutine SECORD contains a model of a second order linear system with a transfer function of the form

$$\frac{\text{OUTPUT(S)}}{\text{INPUT(S)}} = \frac{\omega_n^2}{S^2 + 2\xi\omega_n S + \omega_n^2} \quad . \qquad \text{(IV-4)}$$

The calling statement is of the form

CALL SECORD (INPUT, OUTPUT, X, DX, INDEX, ETA, WN)

where ETA = ξ and WN = ω_n [Equation (IV-4)] and INDEX is an integer defining the element of the array X (Section IV) which corresponds to the first state variable of the model for the second order system. Note that two states are required to model a second order system and that INPUT is a real variable in subroutine SECORD.

8. Subroutine LDLAG

Subroutine LDLAG contains a model of a standard lead-lag compensation network with a transfer function of the form:

$$\frac{OUTPUT(S)}{INPUT(S)} = \frac{\tau_1 s + 1}{\tau_2 s + 1} \quad . \qquad \text{(IV-5)}$$

The calling statement is of the form

```
CALL LDLAG (INPUT, OUTPUT, X, DX, INDEX, T1, T2)
```

where INPUT, OUTPUT, X, DX, and INDEX are defined similarly to the definitions given in Section IV and where T1 = τ_1 and T2 = τ_2 [Equation (IV-5)]. Note that one state variable is required to model a lead-lag network and that INPUT is a real variable in subroutine LDLAG.

9. Subroutine GYRO2

Subroutine GYRO2 contains a model of a seeker gyro which can be torqued. The calling statement is of the form

```
CALL GYRO2 (TGY, TGZ, X, DX, INDEX, IX, ITP, ITY, WS) .
```

Consider Figure 6.

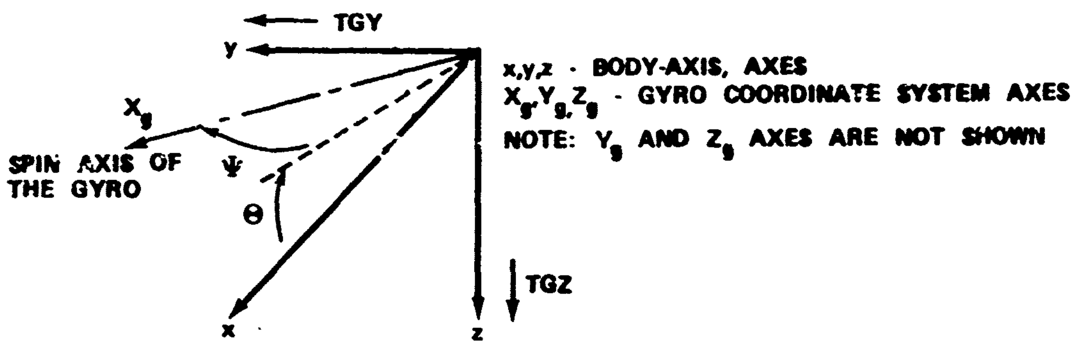


Figure 6. Gyro Coordinate System

Let x, y, and z be the coordinates of the body-axis system (fixed on the missile) and let θ and ψ define the location of the gyro spin axis. Then the arguments of subroutine GYRO2 are defined as follows:

- TGY - Torque on the gyro as defined in Figure 6
- TGZ - Torque on the gyro as defined in Figure 6
- X - Array of state variables [X(INDEX) = θ and X(INDEX + 1) = ψ]
- IX - Axial moment of inertia of the gyro
- ITP - Transverse moment of inertia in the pitch plane
- ITY - Transverse moment of inertia in the yaw plane
- WS - Spin rate of the gyro
- INDEX - An integer defined as in Section IV.

This gyro model does not model nutation of the gyro and thus is suitable for digital integration using a relatively large step size. Note that two states are required to model the gyro. The torques TGY and TGZ are transformed to torques in the gyro coordinate system which has an X_g-axis aligned with the gyro spin axis and which does not rotate or spin with the gyro wheel. The equations used in the model can be obtained from the listing of subroutine GYRO2 given in Appendix C.

10. Subroutine PLOT

Subroutine PLOT was intended to be called in subroutine STRPLT (Section III) and can be used to plot out information using the line printer. Section VI contains an example of a trajectory (altitude versus time) made with subroutine PLOT.

The calling statement is of the form

```
CALL PLOT (A, N, M, NL, NS)
```

where

- A - Matrix of data to be plotted. The first column represents the base variable (e.g., time), and the successive columns are the cross variables (the maximum number of cross variables is nine).
- N - Number of rows in A
- M - Number of columns in A
- NL - Number of lines in plot. If NL = 0 specified, 50 lines are used (one standard line printer page)

NS - Code for sorting the base variable into ascending order. If NS = 0, the base variable is in ascending order and sorting is not required. If NS = 1 sorting is necessary and the base variable will be placed in ascending order.

11. Subroutine TABLE

Subroutine TABLE is an interpolation subroutine which will handle functions with one, two, or three independent variables.

The call statement is of the form

CALL TABLE (N, ANSWER, WT, X, XT, NX, NPX, Y, YT, NY, NPY, Z, ZT, NZ, NPZ)

where

N - The number of independent variables

ANSWER - Dependent variable corresponding to (X, Y, Z)

WT - Table of dependent variables corresponding to (XT, YT, ZT)

X - Independent variable X

XT - Table of independent X values

NX - Number of points in XT

NPX - Number of points to be used for X interpolation

Y - Independent variable Y

YT - Table of independent Y values

NY - Number of points in YT

NYX - Number of points to be used for Y interpolation

Z - Independent variable Z

NZ - Number of points in ZT

NPZ - Number of points to be used for Z interpolation.

The use of subroutine TABLE can be illustrated by a simple example. Assume that Table I is a table of values for the axial drag coefficient C_{D_0}.

The data can be stored in tables by means of a data statements

```
DATA (((CDOT(I,J,K),I=1,5),J=1,4,K=1,2)/
X.8,.7,.6,.4,.3,
X.6,.5,.4,.3,.2,
X.5,.4,.2,.2,.2,
```

```
X,.4,.4,.3,.2,.2,
X,.7,.6,.5,.4,.3,
X,.5,.4,.3,.2,.2,
X.3,.3,.3,.2,.2,
X,.3,.3,.3,.4,.5/
 DATA DEL(I),I=1,4)/-15.,-10.,-5.,0./
 DATA (MACH(I),I=1,2)/.4,.8/
 DATA (ALFAT(I),I=1,5)/0.,5.,10.,15.,20./
```

Table I. Data Table for Drag Coefficient C_{D_0}

Angle of Attack (deg) 0	5	10	15	20	Wing Deflection (deg)	Mach No.
0.8	0.7	0.6	0.4	0.3	-15	0.4
0.6	0.5	0.4	0.3	0.2	-10	0.4
0.5	0.4	0.2	0.2	0.2	-5	0.4
0.4	0.4	0.3	0.2	0.2	0	0.4
0.7	0.6	0.5	0.4	0.3	-15	0.8
0.5	0.4	0.3	0.2	0.2	-10	0.8
0.3	0.3	0.3	0.2	0.2	-5	0.8
0.3	0.3	0.3	0.4	0.5	0	0.8

The program would contain a call statement of the form

```
CALL TABLE(3,CDO,CDOT,ALFA,ALFAT,5,2,DP1,DEL,4,2,MACH,MT,2,2)
```

Thus given ALFA, DP1, and MACH, CDO will be computed by interpolation. Note that NPX, NPY, and NPZ are equal to two. This implies that all interpolation will be linear. Note also that the data tables are arranged in ascending order of magnitude for the independent variables and that the angle of attack table values correspond to the I subscript of the data statement for CDOT(I, J, K). The J and K subscripts correspond to the wing deflection and Mach number table values respectively. When N = 2, the variables Z, ZT, NZ, NPZ are all dummy variables. When N = 1, the variables Y, YT, NY, NYZ, Z, ZT, NZ, NPZ are all dummy variables.

12. Subroutine QSDSUB

Subroutine QSDSUB contains computations for the standard aerodynamic pressure terms. The calling statement is of the form

CALL QSDSUB .

The common block AIR must be included in the subroutine which call QSDSUB. The variables in common block AIR are computed as follows

$$QS = \rho V_m^2 S$$

$$QSD = \rho V_L^2 S D$$

$$QSR = \rho V_m S D^2 .$$

Section V INPUT AND OUTPUT DATA

1. Input Data

The following is a list of input data cards required by the program:

a) SX TX (read in with 2I5 format)

b) TAG(I), I = 1, 8 (read in with 8A10 format)

c) IOPTN, INTOPT (read in with 2I5 format)

d) TIME, PHI, THTA, PSI (read in with 8F10.0 format)

e) U, V, W, X(1), X(2), X(3) (read in with 8F10.0 format)

f) X(11), X(12), X(13) (read in with 8F10.0 format)

g) X(I), I = 14, 13 + SX (read in with 8F10.0 format)

h) X(I), I = 13 + SX + 1, SX + TX + 13 (read in with 8F10.0 format)

i) DT, DTMIN, EMAX, PRNTI, TMAX (read in with 8F10.0 format)

j) MAXPT (read in with 2I5 format)

k) (Blank card)

where

SX is the number of state variables in subroutine SEEKER

TX is the number of state variables in subroutine TARGET

TAG(I), I = 1, 8 is a 80 column title for output (may be a blank card)

IOPTN is the print option (if IOPTN = 0 standard printout; if IOPTN = 1, printout by subroutine WRT; and if IOPTN = 2, standard printout plus printout provided by WRT)

INTOPT is the integration routine option (if INTOPT = 0, Runga-Kutta fourth order used for integration; if INTOPT = 1, variable step Runga-Kutta Merson is used for integration; if INTOPT = 2, Hamming Predictor Corrector is used for integration). See Appendix B for discussion of integration routines.

TIME is the initial value of time

PHI is the initial value of the Euler angle ϕ (Figure 1)

THTA is the initial value of the Euler angle θ (Figure 1)

PSI is the initial value of the Euler angle ψ (Figure 1)

U is the initial value of the missile velocity in the body-axis x-direction

V is the initial value of the missile velocity in the body-axis y-direction

W is the initial value of the missile velocity in the body-axis z-direction

X(1) is the initial value of the body rate p about the body-axis x-axis

X(2) is the initial value of the body rate q about the body-axis y-axis

X(3) is the initial value of the body rate r about the body-axis z-axis

X(11) is the initial displacement of the missile in the earth coordinate system X-direction

X(12) is the initial displacement of the missile in the earth coordinate system Y-direction

X(13) is the initial displacement of the missile in the earth coordinate system Z-direction

X(I), I = 14, 13 + SX are the initial values for the state variables in subroutine SEEKER

X(I), I = 13 + SX + 1, SX + TX + 13 are the initial values for the state variables in subroutine TARGET

DT is the step size to be used for integration

DTMIN is the minimum step size to be used for the variable step size integration methods. The constant has no significance for the fixed step Runga-Kutta integration routine but DTMIN must still be defined.

EMAX is the maximum error to occur in the integration routines. The step size will be reduced until the error is less than EMAX or the minimum step has been reached (DTMIN) (DTMIN used for Runga-Kutta-Merson only)

PRNTI is the print interval in seconds (PRNTI must be greater than or equal to DT)

TMAX is the minimum value for the FORTRAN variable TIME. The run is terminated and the next data set is read in if TIME > TMAX.

MAXPT is the maximum allowable number of printouts. After the run is complete for one data set, the program is initialized, and a new data set is read in. Thus, data sets may be "stacked" in sets. The program can be terminated by setting SX = TX = 0. This can be accomplished by placing a blank card after the last data set.

2. Output Data

a. Standard Common Blocks

The standard common blocks which are part of the main program are

```
COMMON /TRANS/ A(9)
COMMON /DISPL/ U,V,W,DELXE,DELYE,DELZE,XTE,YTE,ZTE,RM
COMMON /ANG/ALFA,BETA,SIGY,SIGZ,DP1,DP2,DP3,DP4
COMMON /AIR/ MACH,VM,VS,QS,QSD,QSR,RHO
COMMON /FRMO / FX,FY,FZ,MX,MY,MZ,IX,IZ,IY,IZX,MASS
COMMON /OTHER1/ITERA,DT,DTMIN,EMAX,SMAX,TMAX,PRNTI
COMMON/OTHER2/ THTA,PSI,PHI,KE,S,D,SX,TX,K,G,MAXPT
COMMON /INT/ R(4)
COMMON/MIS/  RMIN
```

These common blocks may be used to transfer variables from the main program to various user supplied subprograms (and vice versa) and from user supplied subroutine to user supplied subroutine. The variables given in the COMMON statements are defined in Section II.

The following variables must be declared to be real when a common block containing them is used: MX, MY, MZ, IX, IY, IZX, MASS, KE, and K. The variables SX and TX must be declared to be integer variables when common block OTHER2 is used.

b. Output Options

The options available for outputing have been covered in Section II. A sample of the standard printout is given in Section VI. Note that Euler angles are provided in the standard printout. These angles are computed only at the print interval from the quaternions (Appendix A).

Section VI EXAMPLE PROGRAM AND OUTPUT

1. Subprograms Supplied by User

The following listing is comprised of the listing from examples used in the previous sections.

```
SUBROUTINE SEEKER(TIME,X,DX)
REAL INPUT
REAL KE
REAL K,KN
DIMENSION X(1),DX(1)
COMMON /ANG/ALFA,BETA,SIGY,SIGZ,DP1,DY2,DP3,DY4
KN=8.5
K=10.
DX(14)=K*(SIGY-X(14))
DX(15)=K*(SIGZ-X(15))
DP1=KN*DX(14)
DY2=KN*DX(15)
IF(ABS(DP1).GT..1745)  DP1=SIGN(.1745,DP1)
IF(ABS(DY2).GT..1745)  DY2=SIGN(.1745,DY2)
IF(TIME.LT.50.)  DP1=DY2=0.
DY4=DY2
DP3=DP1
RETURN
END
```

```
SUBROUTINE FORGM(TIME,X,DX)
INTEGER SX,TX
REAL KE,K
REAL MACH,MX,MY,MZ,IX,IY,IZ,IZX,MASS
DIMENSION X(1),DX(1)
COMMON /ANG/ALFA,BETA,SIGY,SIGZ,DP1,DY2,DP3,DY4
COMMON /OTHER2/ THTA,PSI,PHI,KE,S,D,SX,TX,K,G,MAXPT
COMMON /AIR/ MACH,VM,VS,QS,QSD,QSR,RHO
COMMON/FRMO/ FX,FY,FZ,MX,MY,MZ,IX,IY,IZ,IZX,MASS
IY=IZ=4.77
IX=.202
MASS=4.671
IZX=0.
S=.1965
D=.5
CDO=.25
CMA=-23.
CNA=15.
```

```
CMD=18.
CMQ=-283.
CALL OSDSUB
FX=-.5*QS*CDO
FY=-.5*QS*CNA*BETA
FZ=-.5*QS*CNA*ALFA
Q=X(2)
R=X(3)
MX=0.
MY=.5*QSD*(CMA*ALFA+CMD*DP1+D*CMQ*Q/(2.*VM))
MZ=.5*QSD*(-CMA*BETA+CMD*DY2+D*CMQ*R/(2.*VM))
RETURN
END
```

```
      SUBROUTINE STRPLT(TIME,X,DX,IMISS,IPLOT)
      DIMENSION X(1),DX(1)
      DIMENSION A(1000),B(500)
      COMMON/OTHER1/ ITERA,DT,DTMIN,EMAX,N,SMAX,TMAX,PRNTI
4     FORMAT(1H1,50X,*ZE VS TIME*// )
      IF(ITERA.NE.1)  GO TO 1
      I=0
      TS=0.
      STEP=1.
1     CONTINUE
      IF(TIME.LT.TS.AND.IPLOT.EQ.0)  GO TO 3
      TS=TIME+STEP
      I=I+1
      A(I)=TIME
      B(I)=X(13)
      IF(IPLOT.EQ.0)  GO TO 3
      WRITE(6,4)
      N=I
      DO 5 J=1,N
5     A(J+N)=B(J)
      CALL PLOT(A,N,2,0,0)
3     RETURN
      END
```

```
SUBROUTINE TARGET(TIME,X,DX)
DIMENSION X(1),DX(1)
COMMON /DISPL/ U,V,W,DELXE,DELYE,DELZE,XTE,YTE,ZTE,RM
DX(16)=5.
XTE=46000.
YTE=500.+X(16)
ZTE=0.
RETURN
END
```

```
SUBROUTINE WRT(TIME,X,DX)
DIMENSION X(1),DX(1)
RETURN
END
```

```
SUBROUTINE WIND(TIME,X,DX,U,V,W)
DIMENSION X(1),DX(1)
RETURN
END
```

2. Input Data

The following initial conditions will be assumed:

t_0 = 36.5 sec	Y = 47.78 ft
u = 762 ft/sec	Z = -13,420 ft
v = 0 ft/sec	ϕ_0 = 0 rad
w = 0 ft/sec	θ_0 = -0.379 rad
p = 0 rad/sec	ψ_0 = 0 rad
q = 0 rad/sec	X_{14} = 0 rad
r = 0 rad/sec	X_{15} = 0 rad
X = 31,500 ft	X_{16} = 0 ft

Let the step size be 0.005 second, the print interval be 0.5 second, and let the maximum time (TMAX) be 70 seconds. Because the print interval is 0.5 second there should be no more than 140 printouts (MAXPT = 140). Select the standard printout and Runga-Kutta integration. The input data card set will then be as follows:

```
    5    10          20          30          40           50          60         70        80
    |    |           |           |           |            |           |          |         |
    8    1
TEST PROGRAM FOR 6D TERMINAL HOMING PROGRAM
    0    0
36.5          0.          -.3791       0.
762.          0.           0.           0.          0.          0.
31500.        47.78       -13420.       0.          0.          0.
0.           0.           0.          0.          0.          0.          0.        0.
0.
.005          .003         .00005       2.          70.
  140
```

3. Output

The following is the output result from the subprograms given in Paragraph 1 of Section V and input data given in Paragraph 2 of Section V.

TEST PROGRAM FOR 6D TERMINAL HOMING PROGRAM --

```
STEP SIZE =      .0050    MINIMUM STEP SIZE =   .00500000    MAXIMUM ERROR =   .00005000
MAX TIME =    70.0000     PRINT INTERVAL =     2.0000

                                 TIME =  36.5000 SEC
VM    =  .7620E+03     MACH   =  .7164E+00     XE     =  .3150E+05     YE     =  .4778E+02     ZE     = -.1342E+05
THTA  = -.3791E+00     PSI    = 0.             PHI    = 0.
U     =  .7620E+03     V      = 0.             W      =  .1019E-11     ALFA   =  .2307E-14     BETA   = 0.
DELXE =  .1450E+05     DELYE  =  .4522E+03     DELZE  =  .1342E+05     FX     = -.2271E+02     MX     = 0.
P     = 0.             Q      = 0.             R      = 0.             FY     = -0.            MY     = -.2493E-11
DP1   = 0.             DY2    = 0.             RANGE  =  .1976E+05     FZ     = -.3252E-11     MZ     = 0.
DP3   = 0.             DY4    = 0.             SIGY   = -.7467E+00     SIGZ   = -.3140E-01     KE     =  .1776E-12

                                 TIME =  38.5050 SEC
VM    =  .7781E+03     MACH   =  .7298E+00     XE     =  .3291E+05     YE     =  .4778E+02     ZE     = -.1279E+05
THTA  = -.4561E+00     PSI    = 0.             PHI    = 0.
U     =  .7781E+03     V      = 0.             W      = -.4399E+00     ALFA   = -.5654E-03     BETA   = 0.
DELXE =  .1209E+03     DELYE  =  .4622E+03     DELZE  =  .1279E+05     FX     = -.2410E+02     MX     = 0.
P     = 0.             Q      = -.3480E-01     R      = 0.             FY     = -0.            MY     =  .7793E+00
DP1   = 0.             DY2    = 0.             RANGE  =  .1831E+05     FZ     =  .6175E+00     MZ     = 0.
DP3   = 0.             DY4    = 0.             SIGY   = -.7740E+00     SIGZ   =  .3530E-01     KE     =  .2239E-10

                                 TIME =  40.5050 SEC
VM    =  .7970E+03     MACH   =  .7432E+00     XE     =  .3430E+05     YE     =  .4778E+02     ZE     = -.1205E+05
THTA  = -.5275E+00     PSI    = 0.             PHI    = 0.
U     =  .7970E+03     V      = 0.             W      = -.4142E-01     ALFA   = -.5193E-04     BETA   = 0.
DELXE =  .1170E+05     DELYE  =  .4722E+03     DELZE  =  .1205E+05     FX     = -.2586E+02     MX     = 0.
P     = 0.             Q      = -.3535E-01     R      = 0.             FY     = -0.            MY     =  .2239E+00
DP1   = 0.             DY2    = 0.             RANGE  =  .1680E+05     FZ     =  .8057E-01     MZ     = 0.
DP3   = 0.             DY4    = 0.             SIGY   = -.8001E+00     SIGZ   =  .4834E-01     KE     =  .1782E-10

                                 TIME =  42.5050 SEC
VM    =  .8206E+03     MACH   =  .7650E+00     XE     =  .3567E+05     YE     =  .4778E+02     ZE     = -.1119E+05
THTA  = -.5946E+00     PSI    = 0.             PHI    = 0.
U     =  .8206E+03     V      = 0.             W      =  .6566E-01     ALFA   =  .8002E-04     BETA   = 0.
DELXE =  .1033E+05     DELYE  =  .4822E+03     DELZE  =  .1119E+05     FX     = -.2801E+02     MX     = 0.
P     = 0.             Q      = -.3202E-01     R      = 0.             FY     = -0.            MY     =  .5156E-01
DP1   = 0.             DY2    = 0.             RANGE  =  .1524E+05     FZ     = -.1345E+00     MZ     = 0.
DP3   = 0.             DY4    = 0.             SIGY   = -.8252E+00     SIGZ   =  .4664E-01     KE     =  .1492E-10

                                 TIME =  44.5050 SEC
VM    =  .8457E+03     MACH   =  .7856E+00     XE     =  .3702E+05     YE     =  .4778E+02     ZE     = -.1021E+05
THTA  = -.6570E+00     PSI    = 0.             PHI    = 0.
U     =  .8457E+03     V      = 0.             W      =  .9699E-01     ALFA   =  .1147E-03     BETA   = 0.
DELXE =  .8983E+04     DELYE  =  .4922E+03     DELZE  =  .1021E+05     FX     = -.3079E+02     MX     = 0.
P     = 0.             Q      = -.3062E-01     R      = 0.             FY     = -0.            MY     = -.3723E-02
DP1   = 0.             DY2    = 0.             RANGE  =  .1361E+05     FZ     = -.2116E+00     MZ     = 0.
DP3   = 0.             DY4    = 0.             SIGY   = -.8493E+00     SIGZ   =  .5474E-01     KE     =  .1286E-10

                                 TIME =  46.5050 SEC
VM    =  .8726E+03     MACH   =  .8074E+00     XE     =  .3835E+05     YE     =  .4778E+02     ZE     = -.9123E+04
THTA  = -.7147E+00     PSI    = 0.             PHI    = 0.
U     =  .8726E+03     V      = 0.             W      =  .8098E-01     ALFA   =  .9280E-04     BETA   = 0.
DELXE =  .7654E+04     DELYE  =  .5022E+03     DELZE  =  .9123E+04     FX     = -.3399E+02     MX     = 0.
P     = 0.             Q      = -.2779E-01     R      = 0.             FY     = -0.            MY     =  .8038E-02
DP1   = 0.             DY2    = 0.             RANGE  =  .1192E+05     FZ     = -.1893E+00     MZ     = 0.
DP3   = 0.             DY4    = 0.             SIGY   = -.8727E+00     SIGZ   =  .5552E-01     KE     =  .1181E-10
```

```
                                          TIME =  48.5050 SEC
VM    =  .9008E+03   MACH  =  .8298E+00   XE    =  .3966E+05   YE    =  .4778E+02   ZE    = -.7028E+04
THTA  = -.7603E+00   PSI   =  0.          PHI   =  0.
U     =  .9008E+03   V     =  0.          W     =  .2598E-01   ALFA  =  .8435E-04   BETA  =  0.
DELXE =  .6340E+04   DELYE =  .5122E+03   DELZE =  .7925E+04   FX    = -.3769E+02   MX    =  0.
P     =  0.          Q     = -.2564E-01   R     =  0.          FY    =  0.          MY    =  .5571E-02
DP1   =  0.          DY2   =  0.          RANGE =  .1017E+05   FZ    = -.1908E+00   MZ    =  0.
DP3   =  0.          DY4   =  0.          SIGY  = -.8396E+03   SIGZ  =  .6082E-01   KE    =  .9344E-11

                                          TIME =  50.5050 SEC
                                          XE    =  .4093E+05   YE    =  .5258E+02   ZE    = -.6618E+04
VM    =  .9258E+03   MACH  =  .8488E+00   PHI   = -.1218E+00
THTA  = -.9283E+00   PSI   =  .2239E+00   W     = -.9421E+02   ALFA  = -.1026E+00   BETA  = -.1127E+00
U     =  .9151E+03   V     = -.1036E+03   DELZE =  .6618E+04   FX    = -.7158E+02   MX    =  0.
DELXE =  .5068E+04   DELYE =  .5174E+03   R     = -.5394E+00   FY    =  .2807E+03   MY    =  .7326E+02
P     =  0.          Q     =  .4918E+00   RANGE =  .4351E+04   FZ    =  .2851E+03   MZ    = -.8258E+02
DP1   = -.7996E-01   DY2   =  .8650E-01   SIGY  = -.9173E+00   SIGZ  =  .1818E+00   KE    =  .1077E-07
DP3   = -.7996E-01   DY4   =  .8650E-01

                                          TIME =  52.5050 SEC
VM    =  .9529E+03   MACH  =  .8681E+00   XE    =  .4212E+05   YE    =  .1408E+03   ZE    = -.5148E+04
THTA  = -.9195E+00   PSI   =  .1214E+00   PHI   = -.9143E-01
U     =  .9528E+03   V     = -.9826E+01   W     = -.2481E+01   ALFA  = -.2884E-02   BETA  = -.9908E-02
DELXE =  .3880E+04   DELYE =  .4392E+03   DELZE =  .5188E+04   FX    = -.4595E+02   MX    =  0.
P     =  0.          Q     = -.8172E-01   R     =  .5544E-01   FY    =  .2752E+02   MY    = -.1284E+02
DP1   = -.1094E-01   DY2   =  .2258E-01   RANGE =  .6478E+04   FZ    =  .7179E+01   MZ    =  .1583E+02
DP3   = -.1094E-01   DY4   =  .2258E-01   SIGY  = -.9268E+00   SIGZ  =  .1127E+00   KE    =  .3145E-09

                                          TIME =  54.5050 SEC
                                          XE    =  .4326E+05   YE    =  .2736E+03   ZE    = -.3612E+04
VM    =  .9833E+03   MACH  =  .8919E+00   PHI   = -.8208E-01
THTA  = -.9100E+00   PSI   =  .1220E+00   W     =  .3003E+02   ALFA  =  .3894E-01   BETA  = -.3539E-02
U     =  .9828E+03   V     = -.3478E+01   DELZE =  .3612E+04   FX    = -.5120E+02   MX    =  0.
DELXE =  .2736E+04   DELYE =  .3168E+03   R     = -.2069E-01   FY    =  .1887E+02   MY    = -.1940E+01
P     =  0.          Q     =  .2343E-01   RANGE =  .4842E+04   FZ    = -.8384E+02   MZ    =  .1814E+01
DP1   =  .3807E-01   DY2   =  .5426E-02   SIGY  = -.9228E+00   SIGZ  =  .1152E+00   KE    = -.2451E-11
DP3   =  .3807E-01   DY4   =  .5426E-02

                                          TIME =  56.5050 SEC
                                          XE    =  .4444E+05   YE    = -.4174E+03   ZE    = -.2008E+04
VM    =  .1010E+04   MACH  =  .9108E+00   PHI   = -.9351E-01
THTA  = -.8779E+00   PSI   =  .1239E+00   W     =  .4460E+02   ALFA  =  .4612E-01   BETA  = -.5183E-02
U     =  .1009E+04   V     = -.5147E+01   DELZE =  .2008E+04   FX    = -.5647E+02   MX    =  0.
DELXE =  .1560E+04   DELYE =  .1828E+03   R     =  .1872E-02   FY    =  .1729E+02   MY    = -.2352E+00
P     =  0.          Q     =  .1444E-01   RANGE =  .2549E+04   FZ    = -.1364E+03   MZ    =  .4156E+00
DP1   =  .5893E-01   DY2   =  .6732E-02   SIGY  = -.9104E+00   SIGZ  =  .1183E+00   KE    =  .4378E-11
DP3   =  .5893E-01   DY4   =  .6732E-02

                                          TIME =  58.5050 SEC
                                          XE    =  .4568E+05   YE    =  .5725E+03   ZE    = -.3955E+03
VM    =  .1032E+04   MACH  =  .9258E+00   PHI   = -.8605E-01
THTA  = -.8525E+00   PSI   =  .1272E+00   W     =  .4455E+02   ALFA  =  .4318E-01   BETA  = -.5292E-02
U     =  .1031E+04   V     = -.5418E+01   DELZE =  .3955E+03   FX    = -.6159E+02   MX    =  0.
DELXE =  .3180E+03   DELYE =  .3753E+02   R     =  .2524E-02   FY    =  .1941E+02   MY    = -.3167E-01
P     =  0.          Q     =  .9606E-02   RANGE =  .5077E+03   FZ    = -.1598E+03   MZ    =  .2047E-01
DP1   =  .5523E-01   DY2   =  .6753E-02   SIGY  = -.8967E+00   SIGZ  =  .1202E+00   KE    =  .2593E-11
DP3   =  .5523E-01   DY4   =  .6753E-02

X X X X X X X X X X X X X X X X X X X X X X X X X X X  EARTH IMPACT POINT X X X X X X X X X X X X X X X X X

DISTANCE FROM TARGET AT EARTH IMPACT =  .6729E-01 FEET
 XE=  .4600E+05     YF =  .6125E+03     ZE =  .5225E-01
```

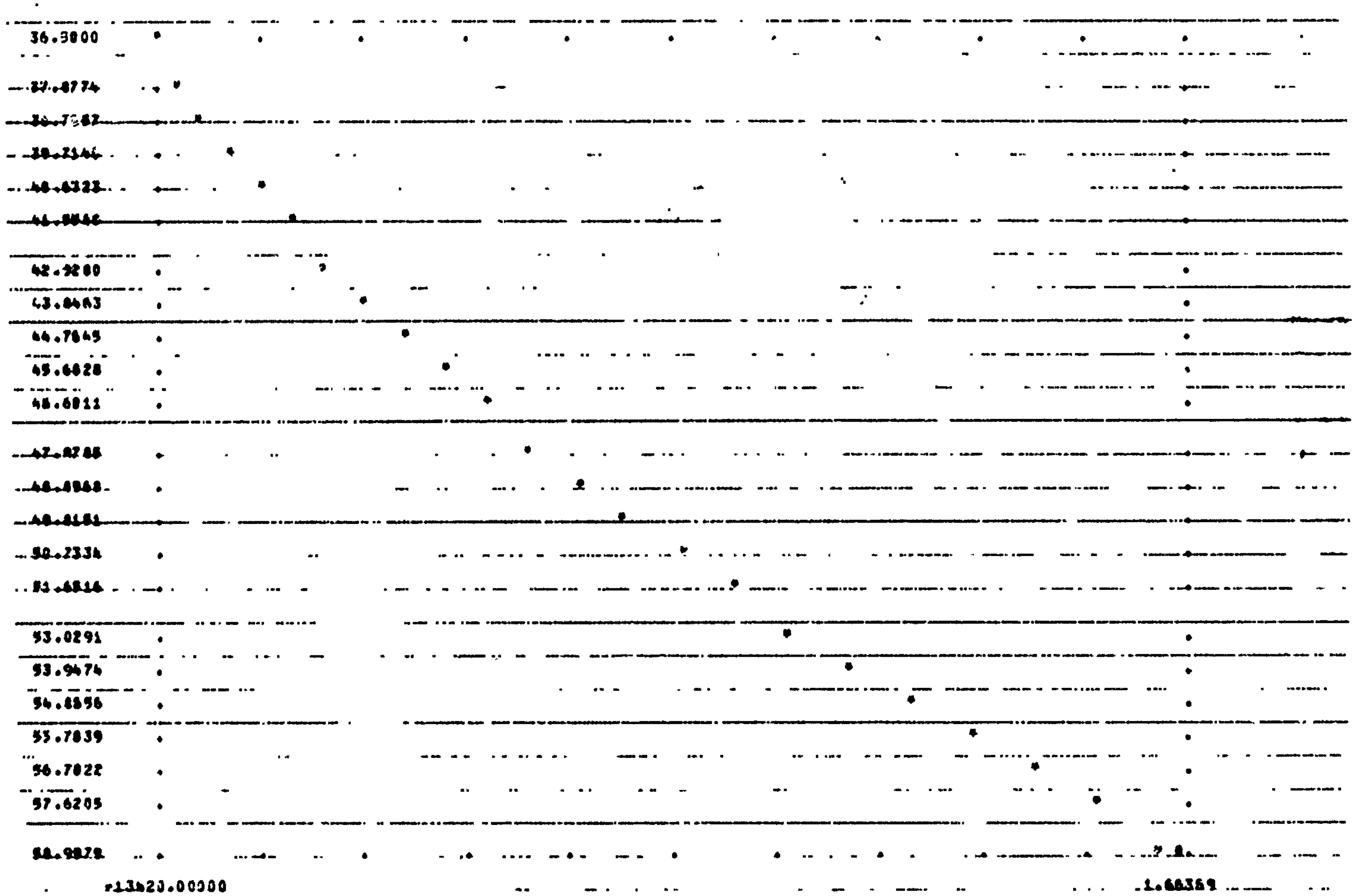
```
+ + + + + + + + + + + + + + + + + + + + + + + MISS DISTANCE + + + + + + + + + + + + + + + + + + + + +

CLOSEST APPROACH TO THE TARGET =  .3640E-01 FEET

DELXE =  .2280E-01     DELYE =  .1853E-02     DELZE =  .2832E-01     (AT POINT OF CLOSEST APPROACH)

+ + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
```

ZE VS TIME



HORIZONTAL AXIS RESOLUTION + OR - 134.2166
VERTICAL AXIS RESOLUTION + OR - .4591

Appendix A. QUATERNIONS

1. Introduction

There are currently three methods in general use for generating coordinate transformations used in 6 degrees of freedom flight simulations: Euler angle rotations, direction cosines, and quaternions. This appendix will deal with quaternions and Euler angle rotations.

2. Euler Angle Rotations

Euler's theorem states that any real rotation may be expressed as a rotation through some angle about some fixed axis. That is, regardless of what the rotational history of the body is, once it reaches some orientation, that orientation may be specified in terms of a rotation (through some angle which can be determined) about some fixed axis. The theorem can be restated in terms of matrices as follows: for every orthogonal transformation matrix $\underline{A}$, there exists some vector $\vec{R}$ such that

$$\underline{A}\,\vec{R} = \vec{R} \quad . \tag{A-1}$$

Equation (A-1) is a statement that there exists a vector coincident with the axis of rotation that is not changed in magnitude or direction for every orthogonal transformation matrix $\underline{A}$. That is, the existence of an axis of rotation can be established for any orthogonal transformation matrix by proving Equation (A-1). The proof of Equation (A-1) can be based on the more general equation

$$\underline{A}\,\vec{R} = \lambda\,\vec{R} \tag{A-2}$$

where λ is a scalar quantity called an eigenvalue. The eigenvalue problem is well known, and it can be shown that the characteristic or secular equation for real orthogonal matrix must have a root $\lambda = +1$.[2] Euler's theorem shows that it is possible to express any rotation (or orthogonal transformation) as a single rotation about some axis and, thus, it is possible to make use of the equivalent rotation to specify orientation.

An orthogonal transformation matrix will now be developed using Euler parameters. Consider Figure A-1. X, Y, and Z are the inertial axes and x, y, and z are the body-axis axes which are assumed to be

---

[2] Gantmacher, F. R., Theory of Matrices, Chelsea Publishing Company, 1960, Library of Congress Catalog Card No. 59-11779.

Preceding page blank

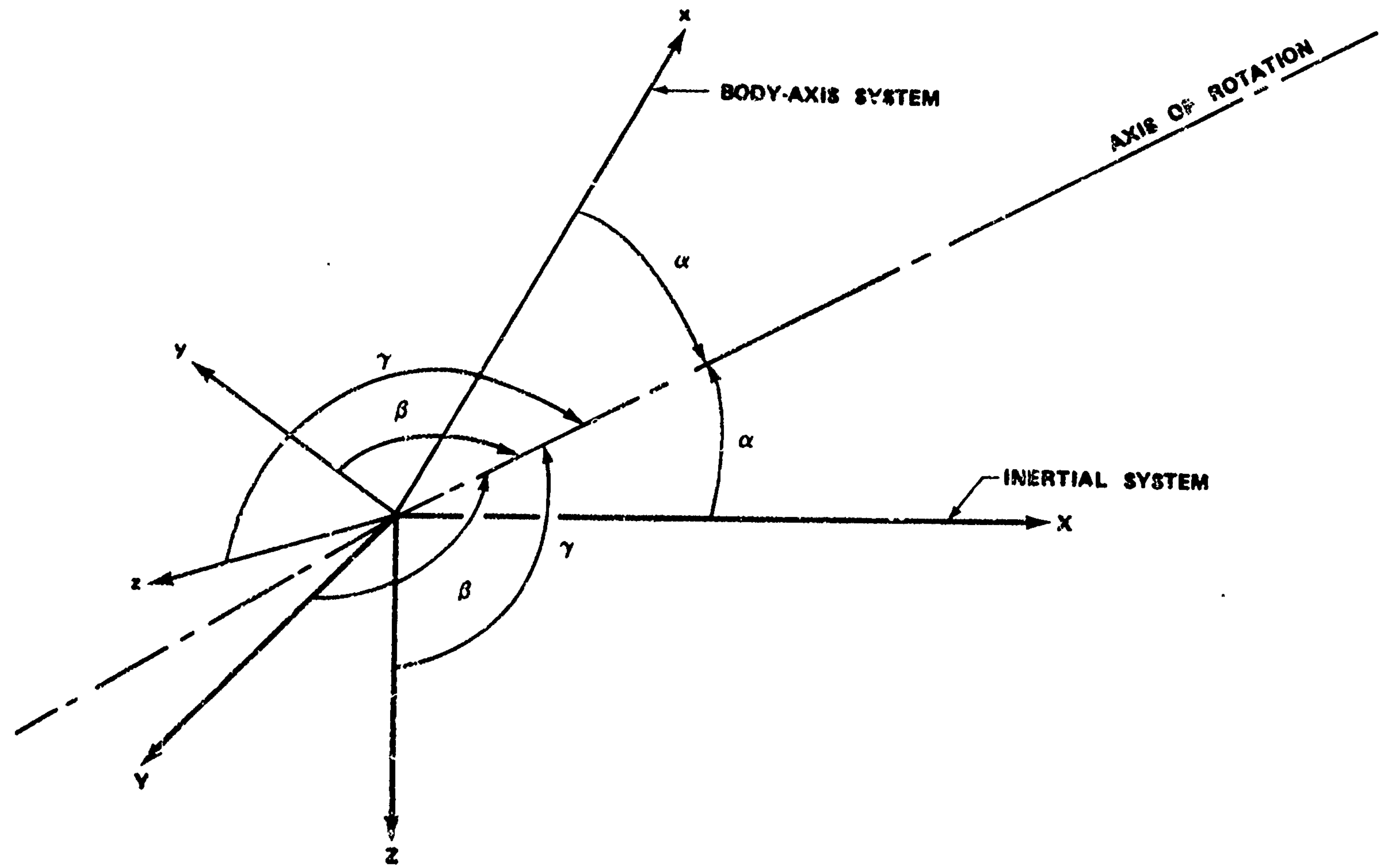


Figure A-1. Euler Axis of Rotation

fixed to a body rotating in space. The task here is to derive the transformation which will take the vector $\vec{I} = (X, Y, Z)$ into the vector $\vec{B} = (x, y, z)$. By Euler's theorem, it is known that an axis of rotation exist (Figure A-1) such that the coordinate system (X, Y, Z) can be "rotated" about the axis and be made to coincide with the (x, y, z) coordinate system for any possible orientation of the (x, y, z) coordinate system. Now let the axis of rotation make angles α, β, and γ with the X, Y, and Z axes, respectively, as shown in Figure A-1. Note it is a geometric consequence that the rotational axis makes the same angles α, β, and γ with the x, y, and z axes, respectively. This fact can be appreciated by the artifice of considering the X, Y, Z coordinate system shown in Figure A-1 to be constructed of wire and welded at the origin to a wire representing the axis of rotation. Clearly, the wire representing the axis of rotation may be rotated until both coordinate systems are coincident. The obvious conclusion is that the angles must be the same. The next step is to define a new coordinate system (X_r, Y_r, Z_r) such that X_r is aligned with the axis of rotation and such that the Y_r axis is in the X-Y plane. There is an orthogonal transformation matrix $\underline{A}$ such that

$$\begin{bmatrix} X_r \\ Y_r \\ Z_r \end{bmatrix} = \begin{bmatrix} \\ \underline{A} \\ \\ \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} . \tag{A-3}$$

In an identical manner, a coordinate system (x_r, y_r, z_r) can be defined for the body-axis coordinate system such that the x_r axis is aligned with the axis of rotation and such that the y_r axis is in the x-y plane. Then

$$\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = \begin{bmatrix} \\ \underline{A} \\ \\ \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{A-4}$$

where the matrix $\underline{A}$ in Equation (A-4) is identical to the matrix $\underline{A}$ in Equation (A-3). Now make the following definitions:

$$\vec{B}_r = \left(x_r, y_r, z_r\right) \tag{A-4a}$$

$$\vec{I}_r = \left(X_r, Y_r, Z_r\right) \tag{A-5}$$

$$\vec{B} = (x, y, z) \tag{A-6}$$

$$\vec{I} = (X, Y, Z) \tag{A-7}$$

then

$$\vec{B}_r = \underline{A}\,\vec{B} \tag{A-8}$$

$$\vec{I}_r = \underline{A}\,\vec{I} \quad . \tag{A-9}$$

Since the x_r and X_r axes coincide

$$\vec{B}_r = \underline{R}\,\vec{I}_r \tag{A-10}$$

where $\underline{R}$ gives a rotation about the axis of rotation and is of the form

$$\underline{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\mu & \sin\mu \\ 0 & -\sin\mu & \cos\mu \end{bmatrix} \tag{A-11}$$

where μ is the amount of angular rotation.

Then using Equations (A-8), (A-9), and (A-10)

$$\vec{B}_r = \underline{R}\,\underline{A}\,\vec{I} \tag{A-12}$$

$$\underline{A}\,\vec{B} = R\,A\,\vec{I} \tag{A-13}$$

$$\vec{B} = \underline{A}^{-1}\,\underline{R}\,\underline{A}\,\vec{I} \quad . \tag{A-14}$$

Thus, an orthogonal transformation from the X, Y, Z coordinate system to the x, y, z coordinate system can be found by finding the matrix A in terms of the angular parameters α, β, and γ. Because the X_r axis is

aligned with the axis of rotation and because the Y_r axis is perpendicular to the Z axis ($a_{23} = 0$), $\underline{A}$ is of the form

$$\underline{A} = \begin{bmatrix} \cos\alpha & \cos\beta & \cos\gamma \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & a_{33} \end{bmatrix} . \qquad \text{(A-15)}$$

Since $\underline{A}$ must be orthogonal matrix, it is possible to deduce

$$\underline{A} = \begin{bmatrix} \cos\alpha & \cos\beta & \cos\gamma \\ \pm\cos\beta\csc\gamma & \pm\cos\alpha\csc\gamma & 0 \\ \pm\cos\alpha\cot\gamma & \pm\cos\beta\cot\gamma & \pm\sin\gamma \end{bmatrix} . \qquad \text{(A-16)}$$

The ambiguities in sign may be removed by geometric considerations with $\alpha = 0$ which implies $\gamma = \beta = \pi/2$. The result is

$$\underline{A} = \begin{bmatrix} \cos\alpha & \cos\beta & \cos\gamma \\ -\cos\beta\csc\gamma & \cos\alpha\csc\gamma & 0 \\ -\cos\alpha\cot\gamma & -\cos\beta\cot\gamma & \sin\gamma \end{bmatrix} . \qquad \text{(A-17)}$$

The desired transformation $\underline{A}^{-1}\,\underline{R}\,\underline{A}$ may now be computed and

$$\underline{A}^{-1}\,\underline{R}\,\underline{A} = \begin{bmatrix} \xi^2 - \eta^2 - \zeta^2 + \chi^2 & 2(\xi\eta + \zeta\chi) & 2(\xi\zeta - \eta\chi) \\ 2(\xi\eta - \zeta\chi) & -\xi^2 + \eta^2 - \zeta^2 + \chi^2 & 2(\eta\zeta + \xi\chi) \\ 2(\xi\zeta + \eta\chi) & 2(\eta\zeta - \xi\chi) & -\xi^2 - \eta^2 + \zeta^2 + \chi^2 \end{bmatrix}$$

(A-18)

where

$$\xi = \cos\alpha \sin\mu/2 \qquad \eta = \cos\beta \sin\mu/2$$

$$\zeta = \cos\gamma \sin\mu/2 \qquad \chi = \cos\mu/2 \quad . \qquad \text{(A-19)}$$

The four parameters ξ, η, ζ, and χ are known as the Euler parameters. It should be noted that the four parameters are not independent because

$$\xi^2 + \zeta^2 + \eta^2 + \chi^2 = 1 \quad . \tag{A-20}$$

Given ξ, ζ, η, and χ it is possible to compute a unique transformation matrix. It is also possible to solve for the Euler parameters given the transformation matrix, but several correct solutions may exist, that is, the solution is not unique.

3. Quaternion Parameters

Quaternions is another four-parameter method to generate orthogonal transformations and is the method used by the computer program. The quaternion q is by definition of the form

$$q = e_0 + e_1 i + e_2 j + e_3 k \tag{A-21}$$

where e_0, e_1, e_2, and e_3 are real numbers and the vector indices i, j, and k are defined by

$$\begin{aligned} i^2 &= -1 \qquad & ij &= -ji = k \\ j^2 &= -1 \qquad & jk &= -kj = i \\ k^2 &= -1 \qquad & ki &= -ik = j \quad . \end{aligned} \tag{A-22}$$

The conjugate of q is defined to be

$$q^* = e_0 - ie_1 - je_2 - ke_3 \quad . \tag{A-23}$$

It can be shown from the definitions above that

$$qq^* = q^* q = e_0^2 + e_1^2 + e_2^2 + e_3^2 \quad . \tag{A-24}$$

If $qq^* = 1$, then q is known as a versor.

The quantity e_0 is called the real or scalar part and $ie_1 + je_2 + ke_3$ is called the complex or vector part. Now let V be a quaternion

whose scalar part is zero. Thus V may be thought of as a vector

$$V = iX + jY + kZ \quad . \tag{A-25}$$

Now consider

$$q^* \, V \, q = \vec{V}' \tag{A-26}$$

where q is a versor ($q^* q = qq^* = 1$).

It can be shown that $\vec{V}'$ is a [illegible] by using the definition established for quaternions, and

$$\vec{V}' = \left(e_0 - ie_1 - je_2 - ke_3\right)\left(iX + jY + kZ\right)\left(e_0 + ie_1 + je_2 + ke_3\right) \tag{A-27}$$

expanding

$$\begin{aligned}
\vec{V}' = {} & i\left\{X\left[e_0 + e_1^2 - e_2^2 - e_3^2\right] + Y\left[2e_3e_0 + 2e_1e_2\right] + Z\left[2e_1e_3 - 2e_0e_2\right]\right\} \\
& + j\left\{X\left[2e_1e_2 - 2e_3e_0\right] + Y\left[e_0^2 - e_1^2 + e_2^2 - e_3^2\right] + Z\left[2e_1e_0 + 2e_3e_2\right]\right\} \\
& + k\left\{X\left[2e_0e_2 + 2e_1e_3\right] + Y\left[2e_1e_3 - 2e_0e_1\right] + Z\left[e_0^2 - e_1^2 - e_2^2 + e_3^2\right]\right\}
\end{aligned} \tag{A-28}$$

or

$$\vec{V}' = \begin{bmatrix} e_0^2+e_1^2-e_2^2-e_3^2 & 2\left(e_3e_0+e_1e_2\right) & 2\left(e_1e_3-e_0e_2\right) \\ 2\left(e_1e_2-e_3e_0\right) & e_0^2-e_1^2+e_2^2-e_3^2 & 2\left(e_1e_0+e_3e_2\right) \\ 2\left(e_1e_3+e_0e_2\right) & 2\left(e_2e_3-e_0e_1\right) & e_0^2-e_1^2-e_2^2+e_3^2 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} . \tag{A-29}$$

The above matrix can be shown to be an orthogonal transformation matrix. Note the 3X3 transformation matrix is completely defined by the quaternion q.

The above matrix is used to transform vectors from the inertial system to the missile body-axis system. There is a one-to-one correspondence between the matrix in Equation (A-18) and the quaternion q. The following relationships may be established by comparing Equations (A-18) and (A-29)

$$e_0 = \chi \qquad e_1 = \xi \qquad e_2 = \eta \qquad e_3 = \zeta \quad .$$

This correspondence shows that the set of all matrices of the form of Equation (A-29) is the same as the set of all transformation matrices obtained by Euler angle rotations.

The standard Euler angle θ, ψ, ϕ (Figure 1) can be computed from the quaternions by the following relationships:

$$\theta = \sin^{-1}\left(-2\left(e_1 e_3 - e_0 e_2\right)\right) \tag{A-30}$$

$$\psi = \tan^{-1}\left(\frac{2\left(e_1 e_2 + e_0 e_3\right)}{2\left(e_0^2 + e_1^2\right) - 1}\right) \tag{A-31}$$

$$\phi = \tan^{-1}\left(\frac{2\left(e_2 e_3 + e_0 e_1\right)}{2\left(e_0^2 + e_3^2\right) - 1}\right) \tag{A-32}$$

which can be established by comparing the transformation matrix using quaternions to the well known three parameter Euler angle transformation matrix. Similarly,

$$e_0 = \cos\psi/2 \cos\theta/2 \cos\phi/2 + \sin\psi/2 \sin\theta/2 \sin\phi/2$$

$$e_1 = \cos\psi/2 \cos\theta/2 \sin\phi/2 - \sin\psi/2 \sin\theta/2 \cos\phi/2$$

$$e_2 = \cos\psi/2 \sin\theta/2 \cos\phi/2 + \sin\psi/2 \cos\theta/2 \sin\phi/2$$

$$e_3 = -\cos\psi/2 \sin\theta/2 \sin\phi/2 + \sin\psi/2 \cos\theta/2 \cos\phi/2 \quad . \tag{A-33}$$

4. Cayley-Klein Parameters

This section will lay the groundwork for the next section where the relationship between the angular rates p, q, and r and the

quaternions will be developed. The basic idea of the Cayley-Klein parameters is to represent a real rotation (or transformation) with a 2X2 complex matrix instead of the usual 3X3 real matrix. Thus, analytic operations are greatly simplified. Consider the following complex matrix

$$\underline{H} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \quad . \tag{A-34}$$

Let the matrix have the following properties:

a) $\underline{H}$ is a unitary matrix

b) $|\underline{H}| = +1.$

Then from the unitary property

$$\underline{H}^* \underline{H} = \underline{I} = \underline{H}\,\underline{H}^* \quad . \tag{A-35}$$

Thus

$$\begin{bmatrix} h_{11}^* & h_{21}^* \\ h_{12}^* & h_{22}^* \end{bmatrix} \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} . \tag{A-36}$$

Expanding and equating components

$$h_{11}^* h_{11} + h_{21}^* h_{21} = 1 \tag{A-37}$$

$$h_{11}^* h_{12} + h_{21}^* h_{22} = 0 \tag{A-38}$$

$$h_{12}^* h_{11} + h_{22}^* h_{21} = 0 \tag{A-39}$$

$$h_{12}^* h_{12} + h_{22}^* h_{22} = 1 \quad . \tag{A-40}$$

Note that the left hand side of Equations (A-38) and (A-39) are the complex conjugates of each other, thus, there are only 3 independent equations. Now from the fact that $|\underline{H}| = +1$, we have

$$h_{11}\, h_{22} - h_{21}\, h_{12} = +1 \quad . \tag{A-41}$$

From Equation (A-38), we have

$$\frac{h_{11}^*}{h_{21}^*} = -\frac{h_{22}}{h_{12}} \quad . \tag{A-42}$$

Then using Equations (A-41) and (A-42)

$$\left(h_{11}\, h_{11}^* + h_{21}^*\, h_{21}\right)\left(\frac{-h_{12}}{h_{21}^*}\right) = 1 \quad . \tag{A-43}$$

Since $(h_{11}\, h_{11}^* + h_{21}^*\, h_{21}) = 1$ from Equation (A-37)

$$h_{12} = -h_{21}^* \quad . \tag{A-44}$$

Similarly

$$h_{22} = h_{11}^* \quad . \tag{A-45}$$

So, the matrix $\underline{H}$ may be written

$$\underline{H} = \begin{bmatrix} h_{11} & h_{12} \\ -h_{12}^* & h_{11}^* \end{bmatrix} \quad . \tag{A-46}$$

The quantities h_{11}, h_{12}, h_{21}, and h_{22} are usually referred to as Cayley-Klein parameters and are in general complex numbers which can be defined by the two complex numbers

$$h_{11} = C_0 + iC_1$$

$$h_{12} = C_2 + iC_3 \qquad \text{(A-47)}$$

and the matrix $\underline{H}$ may be written

$$\underline{H} = \begin{bmatrix} C_0 + iC_1 & C_2 + iC_3 \\ -C_2 + iC_3 & C_0 - iC_1 \end{bmatrix} . \qquad \text{(A-48)}$$

Now consider a matrix $\underline{P}$ which has the following form

$$\underline{P} = \begin{bmatrix} z & x - iy \\ x + iy & -z \end{bmatrix} \qquad \text{(A-49)}$$

where x, y, z are real numbers which can be viewed as coordinates or components of a three-dimensional vector (X, Y, Z) in Euclidean space. Since $\underline{P}$ is Hermitian, the transpose of the complex conjugate of $\underline{P}$ is equal to the matrix $\underline{P}$ and

$$\underline{P}^* = \underline{P} \quad . \qquad \text{(A-50)}$$

Now, consider the similarity transformation of $\underline{P}$ of the form

$$\underline{P}' = \underline{H}\,\underline{P}\,\underline{H}^{-1} \quad . \qquad \text{(A-51)}$$

The following properties of $\underline{P}$ are invarient under a similarity transformation: Hermitian property, trace, and determinant. It follows $\underline{P}'$ must have the following form

$$P' = \begin{bmatrix} z' & x' - iy' \\ x' + iy' & -z' \end{bmatrix} . \qquad \text{(A-52)}$$

Since

$$|\underline{P}'| = |\underline{P}| \quad , \qquad \text{(A-53)}$$

it follows

$$x^2 + y^2 + z^2 = x'^2 + y'^2 + z'^2 \quad . \tag{A-54}$$

If x, y, and z are components of a vector, then the length of the vector has not been changed by the similarity transformation. Then from Equations (A-49), (A-51), and (A-48)

$$\begin{bmatrix} z' & x' - iy' \\ x' + iy' & -z' \end{bmatrix} = \begin{bmatrix} C_0 + iC_1 & C_2 + iC_3 \\ -C_2 + iC_3 & C_0 - iC_1 \end{bmatrix} \begin{bmatrix} z & x - iy \\ x + iy & -z \end{bmatrix} \begin{bmatrix} C_0 - iC_1 & -C_2 - iC_3 \\ C_2 - iC_3 & C_0 + iC_1 \end{bmatrix} \tag{A-55}$$

Then from Equation (A-55), it can be shown

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} C_0^2 - C_1^2 - C_2^2 + C_3^2 & 2(C_0C_1 + C_2C_3) & 2(C_1C_3 - C_0C_2) \\ 2(C_2C_3 - C_0C_1) & C_0^2 - C_1^2 + C_2^2 - C_3^2 & 2(C_1C_2 + C_3C_0) \\ 2(C_0C_2 + C_1C_3) & 2(C_1C_2 - C_0C_3) & C_0^2 + C_1^2 - C_2^2 - C_3^2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{A-56}$$

Using matrix notation

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \underline{A} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad . \tag{A-57}$$

The matrix $\underline{A}$ satisfies orthgonality conditions. The parameters C_0, C_1, C_2, and C_3 may be related to results in the previous two sections by comparing Equations (A-56), (A-29), and (A-18).

$$\begin{aligned} C_0 &= \chi = e_0 \\ C_1 &= \zeta = e_3 \\ C_2 &= \eta = e_2 \\ C_3 &= \xi = e_1 \end{aligned} \tag{A-58}$$

Thus, an equivalence has been indicated between the real 3X3 matrix $\underline{A}$ and the complex 2X2 matrix $\underline{H}$.

It will now be shown that the multiplication of two real 3X3 matrices corresponds to multiplication of two associated 2X2 complex matrices. Consider the read transformation by the 3X3 matrix $\underline{B}$

$$\vec{r}' = \underline{B}\,\vec{r} \quad . \tag{A-59}$$

Now let the associated 2X2 complex matrix be $\underline{H}_1$, so that

$$\underline{P} = \underline{H}_1\, \underline{R}\, \underline{H}_1^{-1} \quad . \tag{A-60}$$

Consider a second transformation $\underline{A}$ associated with $\underline{H}_2$

$$\vec{r}'' = \underline{A}\,\vec{r}$$

$$\underline{P}'' = \underline{H}_2\, \underline{P}\, \underline{H}_2^{-1} \quad . \tag{A-61}$$

Now substitute Equations (A-59) and (A-60) into Equation (A-61)

$$\vec{r}'' = \underline{A}\,\underline{B}\,\vec{r} \tag{A-62}$$

$$\underline{P}'' = \underline{H}_2\, \underline{H}_1\, \underline{P}\, \underline{H}_1^{-1}\, \underline{H}_2^{-1} \tag{A-63}$$

but $\underline{A}\,\underline{B} = \underline{C}$ and $\underline{H}_2\,\underline{H}_1 = \underline{H}_3$ where $\underline{C}$ is a real 3X3 matrix and $\underline{H}_3$ is a complex 2X2 matrix having the form of Equation (A-46). Thus, the multiplication of two real 3X3 matrices corresponds to the multplication of two associated complex matrices in the same order. Thus, there exists a group isomorphism between the multiplicative group of 2X2 matrices of the form of $\underline{H}$ above and the 3X3 real orthogonal matrix.

The 2X2 complex matrix which is associated with a real 3X3 transformation matrix will be used to derive the correspondence between the angular rates p, q, and r and the quaternions e_0, e_1, e_2, e_3 in the next paragraph.

5. Relation Between Quaternions and Angular Rates

The transformation matrix using quaternions has been developed in the preceding section. Since the body-axis angular rates p, q, and r determine the orientation of the body-axis system relative to the inertial system, the relation between the rate of change of the quaternions ($\dot{e}_0$, $\dot{e}_1$, $\dot{e}_2$, $\dot{e}_3$) and the angular rates (p, q, and r) must be established. It was shown in paragraph 4 that an orthogonal transformation matrix may be represented using the Cayley-Klein approach by a 2X2 matrix

$$\underline{H} = \begin{bmatrix} C_0 + iC_1 & C_2 + iC_3 \\ -C_2 + iC_3 & C_0 - iC_1 \end{bmatrix} . \qquad \text{(A-64)}$$

Then using Equations (A-64) and (A-58)

$$\underline{H} = \begin{bmatrix} \cos\frac{\mu}{2} + i\cos\gamma\sin\frac{\mu}{2} & \cos\beta\sin\frac{\mu}{2} + i\cos\alpha\sin\frac{\mu}{2} \\ -\cos\beta\sin\frac{\mu}{2} + i\cos\alpha\sin\frac{\mu}{2} & \cos\frac{\mu}{2} - i\cos\gamma\sin\frac{\mu}{2} \end{bmatrix} . \qquad \text{(A-65)}$$

Now let $\mu = \Delta\mu$ be an infinitesmal rotation. Then if we assume cos $\Delta\mu/2 = 1$ and $\sin\Delta\mu/2 = \Delta\mu/2$,

$$\underline{H}_\epsilon = \begin{bmatrix} 1 + \frac{\Delta\mu}{2}\cos\gamma & \frac{\Delta\mu}{2}\cos\beta + i\frac{\Delta\mu}{2}\cos\alpha \\ -\frac{\Delta\mu}{2}\cos\beta + i\frac{\Delta\mu}{2}\cos\alpha & 1 - i\frac{\Delta\mu}{2}\cos\gamma \end{bmatrix} . \qquad \text{(A-66)}$$

Now assume that the rotation $\Delta\mu$ occurs during the time Δt. If $\underline{H}$ is the matrix at the beginning of the rotation, then $\underline{H}_\epsilon \underline{H}$ is the matrix at the end of the rotation, and the time derivative of $\underline{H}$ may be written

$$\frac{d\underline{H}}{dt} = \lim_{\Delta t \to 0} \left(\frac{\underline{H}_\epsilon \underline{H} - \underline{H}}{\Delta t} \right) = \lim_{\Delta t \to 0} \left(\underline{H}_\epsilon - \underline{I} \right) \underline{H} \qquad \text{(A-67)}$$

then

$$\frac{d\underline{H}}{dt} = \frac{1}{2}\frac{d\mu}{dt}\begin{bmatrix} i\cos\gamma & \cos\beta + i\cos\alpha \\ -\cos\beta + i\cos\alpha & -i\cos\gamma \end{bmatrix}\begin{bmatrix}\underline{H}\end{bmatrix} \quad . \qquad \text{(A-68)}$$

Because $d\mu/dt$ is the scalar quantity of the angular velocity vector, the p, q, and r components of this velocity vector, as defined in Figure 3, are given by

$$p = \frac{d\mu}{dt}\cos\alpha$$

$$q = \frac{d\mu}{dt}\cos\beta$$

$$r = \frac{d\mu}{dt}\cos\gamma \quad . \qquad \text{(A-69)}$$

Therefore

$$\begin{bmatrix} \dot{C}_0 + i\dot{C}_1 & \dot{C}_2 + i\dot{C}_3 \\ -\dot{C}_2 + i\dot{C}_3 & \dot{C}_0 - i\dot{C}_1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} ir & q + ip \\ -q + ip & -ir \end{bmatrix}\begin{bmatrix} C_0 + iC_1 & C_2 + iC_3 \\ -C_2 + iC_3 & C_0 - iC_1 \end{bmatrix} \qquad \text{(A-70)}$$

Expanding and equating like components

$$2\dot{C}_0 = C_3p - C_2q - C_1r$$

$$2\dot{C}_1 = -C_2p + C_3q + C_0r$$

$$2\dot{C}_2 = C_1p + C_0q - C_3r$$

$$2\dot{C}_3 = C_0p - C_1q + C_2r \quad . \qquad \text{(A-71)}$$

Then using Equation (A-58), the relationship between quaternions and angular rates is

$$\dot{e}_0 = -1/2 \left(e_1 p + e_2 q + e_3 r\right)$$

$$\dot{e}_1 = 1/2 \left(e_0 p - e_3 q + e_2 r\right)$$

$$\dot{e}_2 = 1/2 \left(e_3 p + e_0 q - e_1 r\right)$$

$$\dot{e}_3 = 1/2 \left(-e_2 p + e_1 q + e_0 r\right) \quad . \qquad \text{(A-72)}$$

It will be recalled from Paragraph 3 that the quaternion $q = e_0 + ie_1 + je_2 + ke_3$ is a versor which implies

$$e_0^2 + e_1^2 + e_2^2 + e_3^2 = 1 \quad , \qquad \text{(A-73)}$$

Equation (A-72) is used in the computer program to compute the rate of change of the quaternion components subject to the constraint given in Equation (A-73). The matter of a constraint is handled computationally by rewriting Equation (A-72) in the form

$$\dot{e}_0 = -1/2 \left(e_1 p + e_2 q + e_3 r\right) + K \epsilon e_0$$

$$\dot{e}_1 = 1/2 \left(e_0 p - e_3 q + e_2 r\right) + K \epsilon e_1$$

$$\dot{e}_2 = 1/2 \left(e_3 p + e_0 q - e_1 r\right) + K \epsilon e_2$$

$$\dot{e}_3 = 1/2 \left(-e_2 p + e_1 q + e_0 r\right) + K \epsilon e_3 \qquad \text{(A-74)}$$

where

$$\epsilon = 1 - \left(e_0^2 + e_1^2 + e_2^2 + e_3^2\right) \qquad \text{(A-75)}$$

and where K is an arbitrary real, positive constant. The program uses a value of K = 100 which was found by empirical methods to be satisfactory for all cases tested. The value of K may be modified in the program by reading it in through one of the subroutines supplied by the user. However, large values of K should be avoided because the result will be an unstable solution.

Appendix B. INTEGRATION ROUTINES

1. Introduction

There are three integration routines supplied with the main program: Runga-Kutta, Runga-Kutta-Merson, and Hamming Predictor Corrector. The user selects the desired integration routine by an input data card. The step size for all three routines may be altered by the user at any point during the simulation by redefining the FORTRAN variable DT in a user supplied subroutine which contains the common block OTHER2. It should be noted that all the integration subroutines have identical arguments in the call statements but that some of the FORTRAN variables in the argument list may be dummy variables. The equations to be integrated are generated by the EXTERNAL, SUBROUTINE DESUB (TIME, X, DX).

2. Runga-Kutta Integration

The Runga-Kutta integration routine is a fourth order method. The call statement is of the form

```
CALL RUNGA (TIME, X, DX, R, DT, DTMIN, EMAX, NT, IC, SMAX, DESUB) .
```

The set of equations to be solved is of the form

$$\dot{X}_1 = f_1\left(X_1, X_2, \ldots, X_n; t\right)$$

$$\dot{X}_2 = f_2\left(X_1, X_2, \ldots, X_n; t\right)$$

$$\vdots$$

$$\dot{X}_n = f_n\left(X_1, X_2, \ldots, X_n; t\right) \quad . \qquad \text{(B-1)}$$

Then the equations used in fixed step Runga-Kutta integration are

$$X\left(t_K + 1\right)_i = X\left(t_K\right)_i + 1/6 \left(K_{i1} + 2\,K_{i2} + 2\,K_{i3} + K_{i4}\right)$$

$$i = 1, 2, \ldots, n$$

where

$$K_{i1} = h \cdot f_i\left(X_1\left(t_K\right),\ X_2\left(t_K\right),\ \ldots,\ X_n\left(t_K\right);\ t_K\right)$$

$$K_{i2} = h \cdot f_i\left(X_1\left(t_K\right) + K_{11/2},\ X_2\left(t_K\right) + K_{21/2},\ \ldots,\ X_n\left(t_K\right) + K_{n1/2};\ t_K + h/2\right) \qquad \text{(B-2)}$$

$$K_{i3} = h \cdot f_i\left(X_1\left(t_K\right) + K_{12/2},\ X_2\left(t_K\right) + K_{22/2},\ \ldots,\ X_n\left(t_K\right) + K_{n2/2};\ t_K + h/2\right)$$

$$K_{i4} = h \cdot f_i\left(X_1\left(t_K\right) + K_{13},\ X_2\left(t_K\right) + K_{23},\ \ldots,\ X_n\left(t_K\right) + K_{n3};\ t_K + h\right)$$

where h is the step size and t_k denotes a specific time at step size h intervals ($t_K = K.h$ where K = 1, 2, . . .).

This program has the advantage that it requires less computation than Runga-Kutta-Merson and is faster when there are no fast transients (relative to the other transients) which die out and force the use of a small step size. The program also has the advantage in that the user can determine what stage the computation has progressed to in the subroutine by testing the variable R(4). The variable R(4) has the following significances: R(4) = 1.1 implies K_{i1} is being computed (starting with subroutine SEEKER), R(4) = 2.1 implies K_{i2} is being computed, R(4) = 3.1 implies K_{i3} is being computed, and R(4) = 4.1 implies K_{i4} is being computed.

3. Runga-Kutta-Merson

The Runga-Kutta-Merson method is a variable step size program which in certain situations is potentially faster and more accurate than the other methods.[3] This method is a fourth order method which uses the following equations:

[3] Martens, H. R. A Comparative Study of Digital Integration Methods, Simulation, February 1969.

$$K_{i1} = 1/3\ h\ f_i \left[X_1\left(t_K\right),\ X_2\left(t_K\right),\ \ldots,\ X_n\left(t_K\right);\ t_K\right]$$

$$K_{i2} = 1/3\ h\ f_i \left[X_1\left(t_K\right) + K_{11},\ X_2\left(t_K\right) + K_{21},\ \ldots,\ X_n\left(t_K\right) + K_{n1};\ t_K + h/3\right]$$

$$X_{i3} = 1/3\ h\ f_i \left[X_1\left(t_K\right) + .5\ K_{11},\ + .5\ K_{12}\ ,\ X_2\left(t_K\right) + .5\ K_{21}\right.$$

$$\left. + .5\ K_{22},\ \ldots,\ X_n\left(t_K\right) + .5\ K_{n1} + .5\ K_{n2};\ t_K + h/3\right]$$

$$X_{4i} = 1/3\ h\ f_i \left[X_1\left(t_K\right) + 3/8\ K_{11} + 9/8\ K_{13},\ X_2\left(t_K\right) + 3/8\ K_{21}\right.$$

$$\left. + 9/8\ K_{23},\ \ldots,\ X_n\left(t_K\right) + 3/8\ K_{n1} + 9/8\ K_{n3};\ t_K + h/2\right]$$

$$X_{5i} = 1/3\ h\ f_i \left[X_1\left(t_K\right) + 3/2\ K_{11} - 9/2\ K_{13} + 6\ K_{14},\ X_2\left(t_K\right) + 3/2\ K_{21}\right.$$

$$- 9/2\ K_{23} + 6\ K_{24},\ \ldots,\ X_n\left(t_K\right) + 3/2\ K_{n1} - 9/2\ K_{n3}$$

$$\left. + 6\ K_{n4};\ t_K + h\right] \qquad \text{(B-3)}$$

then

$$X_i(k + 1) = X_i(k) + .5\left(K_{i1} + 4\ K_{i4} + K_{i5}\right)$$

where $i = 1, 2, \ldots, n$.

The estimate of the truncation error is

$$\epsilon_i = \left(K_{i1} - 9/2\ K_{i3} + K_{i4} - 1/2\ K_{i5}\right) / 5\ .$$

The step size DT is changed by the program until SMAX, the maximum element of the set $\{|\epsilon_1|, |\epsilon_2|, \ldots, |\epsilon_n|\}$, is less than EMAX, subject to the restriction that DTMIN $\leq$ DT where DTMIN is the minimum allowable step size. If the computed DT is less than DTMIN, DT is set equal to DTMIN. If SMAX $\leq$ EMAX for three steps in a row the step size DT is doubled, and if SMAX $\geq$ EMAX the step size DT is cut in half (DT $\geq$ DTMIN).

The calling statement for the Runga-Kutta-Merson integration routine is of the form

CALL RKMER(TIME, X, DX, R, DT, DTMIN, EMAX, NT, IC, SMAX, DESUB).

The parameter R(4) has a definition which is very similar to the parameter R(4) in the Runga-Kutta routine. The variable R(4) has the following meaning: R(4) = 1.1 implies K_{i1} is being computed,

R(4) = 2.1 implies K_{i2} is being computed, R(4) = 3.1 implies K_{i3} is being computed, R(4) = 4.1 implies K_{i4} is being computed, and R(4) = 5.1 implies K_{i5} is being computed.

4. Hamming Predictor Corrector Intergration

This integration routine is a modified version of IBM SUBROUTINE HPCG.[4] The subroutine is basically the same except that it was changed to be compatible with the other integration routines.

Hamming's modified predictor corrector method is a fourth order method using 4 preceding points for computation of a new vector of the dependent variable X. A fourth order Runga-Kutta method is used for adjustment of the initial step size and for computation of starting values. The routine automatically adjusts the step size DT, halving or doubling. If the step size DT is halved more than 10 successive times, an error message is generated.

The calling statement is of the form

```
CALL HAMPC (TIME, X, DX, R, DT, DTMIN, EMAX, NT, IC, SMAX, DESUB).
```

---

[4] IBM. System/360 Scientific Subroutine Package (360A-CM-03X) Version III Programmer's Manual, H20-0205-3, 1968.

Appendix C. MAIN PROGRAM LISTING

```
      PROGRAM MAIN(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,TAPE2)
C     THIS PROGRAM IS A GENERAL PURPOSE PROGRAM FOR THE 6D SIMULATION OF
C     TERMINAL HOMING MISSILES
C
C     COORDINATE TRANSFORMATIONS ARE MADE WITH QUATERNIONS
C
C     THE USER MUST SUPPLY THE FOLLOWING SUBROUTINES-
C     SEEKER
C     TARGET
C     FORDM
C     MRT
C     STRFLT
C     WIND
C
C     USER INSTRUCTIONS FOR THIS PROGRAM MAY BE FOUND IN RE-TR-72-16
C     21 SEPTEMBER 1972, US ARMY MISSILE COMMAND, REDSTONE ARSENAL,BY
C     DR. LEWIS G. MINOR
C     DR. LEWIS G. MINOR
C
C
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C *   LIST OF VARIABLES USED IN PROGRAM                                 *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C X(1)=P
C X(2)=Q
C X(3)=R
C X(4)=E0
C X(5)=E1
C X(6)=E2
C X(7)=E3
C X(8)=XE DOT
C X(9)=YE DOT
C X(10)=ZE DOT
C X(11)=XE
C X(12)=YE
C X(13)=ZE
C X(I)   I=14,SX+13   ARE THE SEEKER STATES
C X(J)   J=14+SX,SX+TX+13      ARE THE TARGET STATES
C
C
      DIMENSION X(50),DX(50)
      DIMENSION TAG(8)
      EXTERNAL SESUB
      REAL MACH
      REAL MX,MY,MZ,K,KE
      REAL MASS
      REAL IX,IY,IZ,IZX
      INTEGER SX,TX
      COMMON /TRANS/ A(9)
      COMMON /DISPL/ U,V,W,DELXE,DELYE,DELZE,XTE,YTE,ZTE,RM
      COMMON /ANG/ALFA,BETA,SIGY,SIGZ,DP1,DY2,DP3,DY4
      COMMON /AIR/ MACH,VM,VS,QS,QSD,QSR,RHO
      COMMON /FRMO / FX,FY,FZ,MX,MY,MZ,IX,IZ,IY,IZX,MASS
      COMMON /OTHER1/ITERA,DT,DTMIN,EMAX,SMAX,TMAX,PRNTI
      COMMON/OTHER2/ THTA,PSI,PHI,KE,S,B,SX,TX,K,G,MAXPT
```

```
      COMMON/INT/ R(4)
      COMMON/MIS/  RMIN
1-    ITERA=1
      RMIN=1000.
      IFF=0
      IMAXPT=0
      SMAX=0.
      IPLOT=0
      IXI=1
C
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C *       READ INITIAL CONDITIONS                                    *
C*    IF IOPTN=0, STANDARD PRINTOUT/ IF IOPTN=1, SPECIAL PRINTOUT/   *
C*    IF IOPTN=2,STANDARD+SPECIAL PRINTOUT/                          *
C*
C*IF INTOPT=0, FIXED STEP RUNGA-KUTTA INTEGRATION USED               *
C*    IF INTOPT=1, RUNGA-KUTTA-MERSON INTERGRATION USED              *
C*    IF INTOPT=2, HAMMING PREDICTOR CORRECTOR INTEGRATION USED      *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      READ(5,201)  SX,TX
      READ(5,800) (TAG(I),I=1,8)
800   FORMAT(8A10)
      IF(SX.EQ.0) CALL EXIT
      READ(5,201) IOPTN,INTOPT
      NSX=SX+13
      NSX1=NSX+1
      NTSX=TX+SX
      NT=SX+TX+13
      READ(5,11)  TIME,PHI,THTA,PSI
      READ(5,11) U,V,W,X(1),X(2),X(3)
      READ(5,11) X(11),X(12),X(13)
      READ(5,11)   (X(I),I=14,NSX)
      READ(5,11)  (X(I),I=NSX1,NT)
C
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C *    READ STEP SIZE, PRINT INTERVAL, AND MAX TIME                  *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      READ(5,11) DT,DTMIN,EMAX,PRNTI,TMAX
      READ(5,201)  MAXPT
C
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C *   OTHER CONSTANTS                                                *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      G=32.17404
      XASTPT=TIME
      K=100.
      IC=0
C
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C *   COMPUTE IC FOR E0,E1,E2,E3,DXE,DYE,DZE                         *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      COSPSI=COS(PSI/2.)
      COSTHT=COS(THTA/2.)
      COSPHI=COS(PHI/2.)
      SINPSI=SIN(PSI/2.)
```

```
      SINTHT=SIN(THTA/2.)
      SINPHI=SIN(PHI/2.)
C E0
      X(4)=COSPSI*COSTHT*COSPHI+SINPSI*SINTHT*SINPHI
C E1
      X(5)=COSPSI*COSTHT*SINPHI-SINPSI*SINTHT*COSPHI
C E2
      X(6)=COSPSI*SINTHT*COSPHI+SINPSI*COSTHT*SINPHI
C E3
      X(7)=-COSPSI*SINTHT*SINPHI+SINPSI*COSTHT*COSPHI
      CALL SFTOEA(A,X(4),X(5),X(6),X(7))
      CALL MULT(X(8),X(9),X(10),A,U,V,W)
      CALL DESUB(TIME,X,DX)
      R(1)=DT
      GO TO 513
C
C* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C *   START INTEGRATION LOOP                                            *
C* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
9     CONTINUE
      ITERA=ITERA+1
      IF(INTOPT.EQ.1)  GO TO 30
      IF(INTOPT.EQ.2)  GO TO 31
      CALL RUNGA(TIME,X,DX,R,DT,DTMIN,EMAX,NT,IC,SMAX,DESUB)
      GO TO 32
30    CALL RKMER(TIME,X,DX,R,DT,DTMIN,EMAX,NT,IC,SMAX,DESUB)
      GO TO 32
31    CALL HAMPC(TIME,X,DX,R,DT,DTMIN,EMAX,NT,IC,SMAX,DESUB)
32    CONTINUE
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C *   PROGRAM PRINT OUT                                                 *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      IF(TIME.LT.XASTPT.AND.ITERA.NE.1)  GO TO 41
513   XASTPT=XASTPT+PRNTI
      IF(IOPTN.EQ.1)  GO TO 603
      IF(ITERA.NE.1)  GO TO 512
      WRITE(6,801) (TAG(I),I=1,8)
801   FORMAT(1H1,20X,8A10///)
      WRITE(6,204)  DT,DTMIN,EMAX,SMAX,PRNTI
512   WRITE(6,13)  TIME
      WRITE(6,14)  VM,MACH,X(11),X(12),X(13)
      THTA=ASIN(-2.*(X(5)*X(7)-X(4)*X(6)))
      PSI=ATAN2(2.*(X(5)*X(6)+X(4)*X(7)),2.*(X(4)*X(4)+X(5)*X(5))-1.)
      PHI=ATAN2(2.*(X(6)*X(7)+X(4)*X(5)),2.*(X(4)*X(4)+X(7)*X(7))-1.)
      WRITE(6,555) THTA,PSI,PHI
      WRITE(6,500) U,V,W,ALFA,BETA
      WRITE(6,16)  DELXE,DELYE,DELZE,FX,MX
      WRITE(6,501) X(1),X(2),X(3),FY,MY
      WRITE(6,502) DP1,DY2,PK,FZ,MZ
      WRITE(6,503) DP3,DY4,SIGY,SIGZ,KE
      IF(IOPTN.EQ.0)  GO TO 701
603   CALL WRT(TIME,X,DX)
701   IMAXPT=IMAXPT+1
      IF(IMAXPT.LE.MAXPT)  GO TO 41
      WRITE(6,780)  MAXPT
```

```
      GO TO 1
41    CONTINUE
      CALL STRPLT(TIME,X,DX,IMISS,IPLOT)
      IF(IPLOT.EQ.0)  GO TO 607
      IF(IMISS) 605,605,606
607   CONTINUE
      IF(TIME.GT.TMAX)  GO TO 1
C* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C*    COMPUTE MISS DISTANCE                                              *
C* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      IF(X(13).LE.0.)  GO TO 612
      IF(IXI.EQ.0.)  GO TO 612
      ROM=SQRT(DELXE**2+DELYE**2+DELZE**2)
      WRITE(6,609)
      WRITE(6,610) ROM,X(11),X(12),X(13)
      WRITE(6,909)
909   FORMAT(//*X X X X X X X X X X X X X X X X X X X X X X X X X X X X X
     1X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X*//)
610   FORMAT(*  DISTANCE FROM TARGET AT EARTH IMPACT =*,E11.4, * FEET*/
     1*   XE=*,E11.4,5X,*YE =*,E11.4,5X,*ZE =*,E11.4//)
609   FORMAT(///*  X X X X X X X X X X X X X X X X X X X X X X X X X X
     1EARTH IMPACT POINT X X X X X X X X X X X X X X X X X X X X X X*//)
      IXI=0
612   IF(RM.GT.50.)  GO TO      INTOPT=0
      IF(DT.GT..0001)  DT=
      IF(RM.GT.RMIN)  IFF=IFF+1
      IF(IFF.GT.20)  GO TO 601
      IF(RMIN.LE.RM)  GO TO 502
      DXMIN=DELXE
      DYMIN=DELYE
      DZMIN=DELZE
      RMIN=RM
      GOTO 602
601   WRITE(6,302)
      WRITE(6,303)  RMIN,DXMIN,DYMIN,DZMIN
      WRITE(6,910)
      IPLOT=1
      IMISS=0
      GO TO 41
605   CONTINUE
      GO TO 1
630   IF(X(13).LT.50.)  GO TO 602
      IPLOT=1
      IMISS=1
      WRITE(6,302)
      WRITE(6,604)
      WRITE(6,910)
      GO TO 41
606   CONTINUE
      GO TO 1
602   CONTINUE
      GO TO 9
604   FORMAT(///1X,34HMISS DISTANCE GREATER THAN 50 FEET        )
302   FORMAT(///1X,123H* * * * * * * * * * * * * * * * * * * * * * * * *
     1MISS DISTANCE* * * * * * * * * * * * * * * * * * * * * * * * *
```

```
303   FORMAT(//1X,32HCLOSEST APPROACH TO THE TARGET =,E11.4,5H FEET/
     1/1X,7HDELXE =E11.4,5X,7HDELYE =,E11.4,5X,7HDELZE =,E11.4,5X,30H(AT
     2 POINT OF CLOSEST APPROACH)
201   FORMAT(2I5)
11    FORMAT(8F10.3)
910   FORMAT(//*+ + + + + + + + + + + + + + + + + + + + + + + + + + + + +
     1+ + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +*//)
700   FORMAT(//4H ***,100X,3H***/35X,51HYOU HAVE EXCEEDED THE MAXIMUM NU
     1MBER OF PRINT OUTS(,I5,1H)/22X,67HTO SAVE THE NATIONAL RESOURCE OF
     2 TREES THIS RUN IS BEING TERMINATED    )
204   FORMAT(//2X,12HSTEP SIZE = ,F10.4,5X,20HMINIMUM STEP SIZE = ,F11.8
     1,5X,16HMAXIMUM ERROR = ,F11.8/1X,11HMAX TIME = ,F10.4,5X,17HPRINT
     2INTERVAL = ,F10.4//)
13    FORMAT(/50X,7HTIME = ,F8.4,4H SEC   )
14    FORMAT(1X,7HVM     =,E11.4,5X,7HMACH  =,E11.4,5X,7HXE     =,E11.4,5X
     1,7HYE     =,E11.4,5X,7HZE     =,E11.4)
500   FORMAT(1X,7HU      =,E11.4,5X,7HV      =,E11.4,5X,7HW      =,E11.4,5X
     1,7HALFA   =,E11.4,5X,7HBETA   =,E11.4)
15    FORMAT(1X,7HDELXE =,E11.4,5X,7HDELYE =,E11.4,5X,7HDELZE =,E11.4,5X
     1,7HFX     =,E11.4,5X,7HMX     =,E11.4)
501   FORMAT(1X,7HP      =,E11.4,5X,7HQ      =,E11.4,5X,7HR      =,E11.4,5X
     1,7HFY     =,E11.4,5X,7HMY     =,E11.4)
502   FORMAT(1X,7HDP1    =,E11.4,5X,7HDY2    =,E11.4,5X,7HRANGE =,E11.4,5X
     1,7HFZ     =,E11.4,5X,7HMZ     =,E11.4)
503   FORMAT(1X,7HDP3    =,E11.4,5X,7HDY4   = E11.4,5X,7HSIGY = E11.4,5X,
     X7HSIGZ  =,E11.4,5X,7HKE     =,E11.4)
505   FORMAT(1X,7HTHTA   =,E11.4,5X,7HPSI    =,E11.4,5X,7HPHI    =,E11.4)
      END
```

```
SUBROUTINE BFTOEA (A,E0,E1,E2,E3)
DIMENSION A(1)
E0S=E0*E0
E1S=E1*E1
E2S=E2*E2
E3S=E3*E3
E12=E1*E2
E03=E0*E3
E13=E1*E3
E02=E0*E2
E23=E2*E3
E01=E0*E1
A(1)=E0S+E1S-E2S-E3S
A(2)=2.*(E12+E03)
A(3)=2.*(E13-E02)
A(4)=2.*(E12-E03)
A(5)=E0S+E2S-E1S-E3S
A(6)=2.*(E23+E01)
A(7)=2.*(E13+E02)
A(8)=2.*(E23-E01)
A(9)=E0S+E3S-E1S-E2S
RETURN
END
```

```
SUBROUTINE EATOBF(AI,E0,E1,E2,E3)
DIMENSION AI(9),A(9)
CALL BFTOEA(A,E0,E1,E2,E3)
AI(1)=A(1)
AI(2)=A(4)
AI(3)=A(7)
AI(4)=A(2)
AI(5)=A(5)
AI(6)=A(8)
AI(7)=A(3)
AI(8)=A(6)
AI(9)=A(9)
RETURN
END
```

```
      SUBROUTINE AIRTAB(H,RHO,VA)
      DIMENSION Y(12),D(12),T(12)
      DATA Y(1),Y(2),Y(3),Y(4),Y(5),Y(6),Y(7),Y(8),Y(9),Y(10),Y(11),
     $Y(12)/0.,11000.,25000.,36000.,50000.,80000.,90000.,105000.,150000.
     $,164000.,197000.,255000./
      DATA D(1),D(2),D(3),D(4),D(5),D(6),D(7),D(8),D(9),D(10),D(11),
     $D(12)/2.378E-3,1.702E-3,1.056E-3,.709E-3,.3622E-3,.0861E-3,
     $ .0535E-3,.0251E-3,.2085E-5,.173E-5,.568E-5,.012E-5/
      DATA T(1),T(2),T(3),T(4),T(5),T(6),T(7),T(8),T(9),T(10),T(11),
     $T(12)/518.67,479.47,429.47,392.07,392.07,392.07,392.07,392.07,
     $ 613.37,629.67,629.67,431.67/
C
      IF(H.LT.Y(1))  H=Y(1)
      IF(H.GT.Y(12))  H=Y(12)
      DO 15 I=1,12
      IF(H.GE.Y(I).AND.H.LT.Y(I+1))  GO TO 14
   15 CONTINUE
   14 PROP= (H - Y(I))/(Y(I+1) - Y(I))
      RHO= D(I) + (D(I+1) - D(I))*PROP
      TEM= T(I) + (T(I+1) - T(I))*PROP
      VA= 49.02055*SQRT(TEM)
      RETURN
      END
```

```
      SUBROUTINE DESUB(TIME,X,DX)
      REAL MACH,KE,IX,IY,IZ,IZX,MASS,K,MX,MY,MZ
      INTEGER SX,TX
      DIMENSION X(1),DX(1)
      COMMON /INT/ R(4)
      COMMON/XXX/VM2
      COMMON /TRANS/ A(9)
      COMMON /DISPL/ U,V,W,DELXE,DELYE,DELZE,XTE,YTE,ZTE,RM
      COMMON /ANG/ALFA,BETA,SIGY,SIGZ,DP1,DY2,DP3,DY4
      COMMON /AIR/ MACH,VM,VS,QS,QSD,QSR,RHO
      COMMON /FRMO / FX,FY,FZ,MX,MY,MZ,IX,IZ,IY,IZX,MASS
      COMMON /OTHER1/ITERA,DT,DTMIN,EMAX,SMAX,TMAX,PRNTI
      COMMON/OTHER2/ THTA,PSI,PHI,KE,S,D,SX,TX,K,G,MAXPT
      CALL EATOBF(A,X(4),X(5),X(6),X(7))
      UA=X(8)
      VA=X(9)
      WA=X(10)
      CALL WIND(TIME,X,DX,UA,VA,WA)
      CALL MULT(U,V,W,A,UA,VA,WA)
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C *   COMPUTE LOS;TARGET MOTION USING EARTH COORDINATE SYSTEM        *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      CALL TARGET(TIME,X,DX)
      DELXE=XTE-X(11)
      DELYE=YTE-X(12)
      DELZE=ZTE-X(13)
      RM=SQRT(DELXE*DELXE+DELYE*DELYE+DELZE*DELZE)
      SIGZ=ATAN2(DELYE,DELXE)
      SIGY=ATAN2(-DELZE,DELXE)
C
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C *   COMPUTE SEEKER DYNAMICS AND BODY FIXED WING COMMANDS            *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      CALL SEEKER(TIME,X,DX)
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C *   COMPUTE RHO AND SPEED OF SOUND AS A FUNTION OF ALTITUDE         *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      H=-X(13)
      CALL AIRTAB(H,RHO,VS)
      VM2=U*U+V*V+W*W
      VM=SQRT(VM2)
      MACH=VM/VS
C
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C *   COMPUTE ANGLE OF ATTACK                                         *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      IF(U.EQ.0.) U=1.E-20
      ALFA=ATAN2(W,U)
      BETA=ATAN2(V,U)
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C *   COMPUTE FORCES AND MOMENTS IN BODY FIXED SYSTEM                 *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      CALL FOROM(TIME,X,DX)
C
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
C * ---DE OF MOTION IN THE BODY FIXED SYSTEM+QUARTERNIONS-- ------------ *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      DHX=HX-X(2)*X(3)*(IZ-IY)-X(2)*X(1)*IZX
      DHY=HY-X(1)*X(3)*(IX-IZ)-(X(1)*X(1)-X(3)*X(3))*IZX
      DHZ=HZ-X(1)*X(2)*(IY-IX)-X(2)*X(3)*IZX
      DEN=IX*IZ-IZX*IZX
C DP
      DX(1)=(DHX*IZ+DHZ*IZX)/DEN
C DQ
      DX(2)=DHY/IY
C DR
      DX(3)=(DHZ*IX+DHX*IZX)/DEN
C CONSTRAINT
      KE=(1.-X(4)*X(4)-X(5)*X(5)-X(6)*X(6)-X(7)*X(7) )*K
C DE0
      DX(4)=-.5*(X(5)*X(1)+X(6)*X(2)+X(7)*X(3))+KE*X(4)
C DE1
      DX(5)=.5*(X(4)*X(1)+X(6)*X(3)-X(7)*X(2))+KE*X(5)
C DE2
      DX(6)=.5*(X(4)*X(2)+X(7)*X(1)-X(5)*X(3))+KE*X(6)
C DE3
      DX(7)=.5*(X(4)*X(3)+X(5)*X(2)-X(6)*X(1))+KE*X(7)
C
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C * - CHANGE FORCES FORM BODY AXIX TO EARTH - -                        *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      CALL BFTOEA(A,X(4),X(5),X(6),X(7))
      CALL MULT(FXE,FYE,FZE,A,FX,FY,FZ)
C ZE DOULBLE DOT
      DX(10)=FZE/MASS+G
C ZE DOT
      DX(13)=X(10)
C YE DOUBLE DOT
      DX(9)=FYE/MASS
C YE DOT
      DX(12)=X(9)
C XE DOUBLE DOT
      DX(8)=FXE/MASS
C XE DOT
      DX(11)=X(8)
      RM=SQRT(DELXE*DELXE+DELYE*DELYE+DELZE*DELZE)
      RETURN
      END
```

```
      SUBROUTINE RUNGA(TIME,V,F,W,DEL,DELMIN,EMAX,N,IC,SMAX,DESUB)
      DIMENSION VS(50),C(4,50)
      DIMENSION V(1),F(1),W(4)
      IF(IC.GT.0)  GO TO 20
13    H=DEL
      IFG=0
      W(1)=H
      W(2)=DEL
      NH=0
      W(3)=NH
      GO TO 105
C     RESTART IF INPUT DEL HAS CHANGED
20    IF(DEL.NE.W(2))  GO TO 13
105   TIME0=TIME
      DO 106  I=1,N
106   VS(I)=V(I)
      IF(IFG.GT.0)  GO TO  120
      W(4)=1.1
      CALL DESUB(TIME,V,F)
      DO 119  J=1,N
119   C(1,J)=F(J)*H
120   TIME=TIME0+H/2.
      DO 130  J=1,N
130   V(J)=VS(J)+C(1,J)/2.
      W(4)=2.1
      CALL DESUB(TIME,V,F)
      DO 140  J=1,N
140   C(2,J)=F(J)*H
      DO 150  J=1,N
150   V(J)=VS(J)+C(2,J)/2.
      W(4)=3.1
      CALL DESUB(TIME,V,F)
      DO 300  J=1,N
300   C(3,J)=F(J)*H
      TIME=TIME0+H
      DO 301 J=1,N
301   V(J)=VS(J)+C(3,J)
      W(4)=4.1
      CALL DESUB(TIME,V,F)
      DO 302  J=1,N
302   C(4,J)=F(J)*H
      DO 303  J=1,N
303   V(J)=VS(J)+(C(1,J)+2.*C(2,J)+2.*C(3,J)+C(4,J))/6.
      W(4)=1.1
      CALL DESUB(TIME,V,F)
      DO 200  J=1,N
200   C(1,J)=F(J)*H
      IFG=1
      IC=1
      RETURN
      END
```

```
SUBROUTINE MULT(OUTX,OUTY,OUTZ,B,INX,INY,INZ)
REAL INX,INY,INZ
DIMENSION B(9)
OUTX=B(1)*INX+B(4)*INY+B(7)*INZ
OUTY=B(2)*INX+B(5)*INY+B(8)*INZ
OUTZ=B(3)*INX+B(6)*INY+B(9)*INZ
RETURN
END
```

```
      SUBROUTINE RKIER(TIME,V,F,W,DEL,DELMIN,EMAX,N,IC,BMAX,DESUB)
      DIMENSION V(1),W(4),C(5,50),VS(50),F(1),E(50)
C          COMPUTE STOP FOR THIS INTEGRATION STEP
3     TSTOP=TIME+DEL
C          CHECK FOR FIRST TIME INTO ROUTINE
   10 IF(IC .GT. 0)   GO TO 20
      EMIN=EMAX/32.
C          FIRST TIME IN
   13 H = DEL
      IFG=0
      NH = 0
      W(2)  = DEL
      GO TO 105
C          RESTART IF INPUT DEL HAS CHANGED
   20 IF(DEL .NE. W(2))   GO TO 13
      H = W(1)
      NH= W(3)
C          INTEGRAT USING R-K
C          SAVE V TABLE VALUES IN VS ARRAY
  105 TIMEO=TIME
      DO 106  I=1,N
  106 VS(I) = V(I)
      IF(IFG.GT.0)   GO TO 120
C          FIRST PASS THRU R-K COMPUTATION
      W(4)=1.1
  110 CALL DESUB (TIME,V,F)
      DO 119  J=1,N
  119 C(1,J)=F(J)*H/3.
  120 TIME=TIMEO+H/3.
      DO 130 J=1,N
  130 V(J)=VS(J)+C(1,J)
      W(4)=2.1
      CALL DESUB(TIME,V,F)
      DO 140  J=1,N
  140 C(2,J)=F(J)*H/3.
      DO 150  J=1,N
  150 V(J)=VS(J)+.5*C(1,J)+.5*C(2,J)
      W(4)=3.1
      CALL DESUB(TIME,V,F)
      DO 300 J=1,N
  300 C(3,J)=F(J)*H/8.
      TIME=TIMEO+H/2.
      DO 301  J=1,N
  301 V(J)=VS(J)+3.*C(1,J)/8.+9.*C(3,J)/8.
      W(4)=4.1
      CALL DESUB(TIME,V,F)
      DO 302  J=1,N
  302 C(4,J)=F(J)*H/3.
      TIME=TIMEO+H
      DO 303  J=1,N
  303 V(J)=VS(J)+1.5*C(1,J)-4.5*C(3,J)+6.*C(4,J)
      W(4)=5.1
      CALL DESUB(TIME,V,F)
      DO 304  J=1,N
  304 C(5,J)=F(J)*H/3.
```

```
      IF(DEL.LE.DELMIN)  GO TO 160
      IF(.5*H.LT.DELMIN)  GO TO 160
C         TEST FOR HALVING H
      SMAX=0.
      DO 153  J=1,N
      E(J)=(C(1,J)-4.5*C(3,J)+4.*C(4,J)-.5*C(5,J))/5.
      ERROR=ABS(E(J))
      IF(ERROR.GT.EMAX)  GO TO 155
      IF(ERROR.GT.SMAX)  SMAX=ERROR
153   CONTINUE
      IF(SMAX.LT.EMIN)  IC=IC+1
      IF(SMAX.GE.EMIN)  IC=1
C     CHECK FOR THIRD ITERATION WITH ERROR LT EMIN
      IF(IC.LE.3)  GO TO 160
      H2=2.*H
      IC=1
      IF(H2.GT.DEL)  GO TO 160
      H=H2
C     FINISH THIS ITERATION
      GO TO 160
C         HALVE H
  155 H = .5*H
      NH = NH+1
      IC = 1
C     START R-K INTEGRATION OVER
      TIME=TIMEO
      DO 157  J=1,N
  157 V(J) = VS(J)
      GO TO 110
C         UPDATE V TABLE
160   DO 180  J=1,N
180   V(J)=VS(J)+.5*(C(1,J)+C(5,J))+2.*C(4,J)
      W(4)=1.1
      CALL DESUB(TIME,V,F)
      DO 200  J=1,N
200   C(1,J)=F(J)*H/3.
      IFG=1
C         END OF REQUIRED INTEGRATION
220   W(1)=H
      W(2) = DEL
      W(3) = NH
C
      RETURN
      END
```

```
      SUBROUTINE HAMPC(X    ,Y,DERY,W,H   ,DELMIN,EMAX,NDIM,IC,SMAX,FCT)
      DIMENSION Y(1),DERY(1),AUX(16,20),W(1)
      COMMON IHLF,  ISTEP
C     CHECK FOR FIRST TIME IN ROUTINE
      IF(IC.GT.0)  GO TO 300
313   W(2)=H
      N=1
      IHLF=0
      TIME=X
      GO TO 305
C     RESTART IF INPUT DEL HAS CHANGED
300   IF(H   .NE.W(2))  GO TO 313
      IF(IC.EQ.2) GO TO 24
      GO TO 211
305   DO 1 I=1,NDIM
      AUX(16,I)=0.
      AUX(15,I)=DERY(I)
1     AUX(1,I)=Y(I)
C
C     COMPUTATION OF DERY FOR STARTING VALUES
C
4     CALL FCT(X,Y,DERY)
7     DO 8 I=1,NDIM
8     AUX(8,I)=DERY(I)
C
C     COMPUTATION OF AUX(2,I)
      ISW=1
      GO TO 100
C
9     X=X+H
      DO 10 I=1,NDIM
10    AUX(2,I)=Y(I)
C
C     INCREMENT H IS TESTED BY BISECTION
11    IHLF=IHLF+1
      X=X-H
      DO 12 I=1,NDIM
12    AUX(4,I)=AUX(2,I)
      H=.5*H
      N=1
      ISW=2
      GO TO 100
C
13    X=X+H
      CALL FCT(X,Y,DERY)
      N=2
      DO 14 I=1,NDIM
      AUX(2,I)=Y(I)
14    AUX(9,I)=DERY(I)
      ISW=3
      GO TO 100
C
C     COMPUTATION OF TEST VALUE DELT
15    DELT=0.
      DO 16 I=1,NDIM
```

```
15    DELT=DELT+AUX(15,I)*ABS(Y(I)-AUX(4,I))
      DELT=.06666667*DELT
      SMAX=DELT
      IF(DELT-EMAX) 19,19,17
17    IF(IHLF-10) 11,18,18
C
C     NO SATISFACTORY ACCURACY AFTER 10 BISECTIONS. ERROR MESSAGE.
18    IHLF=11
      X=X+H
      GO TO 4
C
C     THESE IS SATISFACTORY ACCURACY AFTER LESS THAN 11 BISECTIONS.
19    X=X+H
      CALL FCT(X,Y,DERY)
      DO 20 I=1,NDIM
      AUX(3,I)=Y(I)
20    AUX(10,I)=DERY(I)
      N=3
      ISW=4
      GO TO 100
C
21    N=1
      X=X+H
      CALL FCT(X,Y,DERY)
      X=TIME
      DO 22 I=1,NDIM
      AUX(11,I)=DERY(I)
22    Y(I)=AUX(1,I)+H*(.375*AUX(8,I)+.7916667*AUX(9,I)
     $-.2083333*AUX(10,I)+.04166667*DERY(I))
23    X=X+H
      N=N+1
      CALL FCT(X,Y,DERY)
      W(2)=H
      W(1)=IHLF
      IC=2
      RETURN
24    IF(N-4) 25,200,200
25    DO 26 I=1,NDIM
      AUX(N,I)= Y(I)
26    AUX(N+7,I)=DERY(I)
      IF(N-3) 27,29,200
C
27    DO 28 I=1,NDIM
      DELT=AUX(9,I)+AUX(9,I)
      DELT=DELT+DELT
28    Y(I)=AUX(1,I)+.3333333*H*(AUX(8,I)+DELT+AUX(10,I))
      GO TO 23
C
29    DO 30 I=1,NDIM
      DELT= AUX(9,I)+AUX(10,I)
      DELT=DELT+DELT+DELT
30    Y(I)=AUX(1,I)+.375*H*(AUX(8,I)+DELT+AUX(11,I))
      GO TO 23
C
C     THE FOLLOWING PART OF SUBROUTINE HAMPC COMPUTES BY MEANS OF
```

```
C       RUNGA-KUTTA METHOD STARTING VALUES FOR THE
C       PREDICTOR-CORRECTOR METHOD
100     DO 101 I=1,NDIM
        Z=H*AUX(N+7,I)
        AUX(5,I)=Z
101     Y(I)=AUX(N,I)+.4*Z
C       Z IS AN AUXILIARY STORAGE LOCATION
C
        Z=X+.4*H
        CALL FCT(Z,Y,DERY)
        DO 102 I=1,NDIM
        Z=H*DERY(I)
        AUX(6,I)=Z
102     Y(I)=AUX(N,I)+.2969776*AUX(5,I)+.1587595*Z
C
        Z=X+.4557372*H
        CALL FCT(Z,Y,DERY)
        DO 103 I=1,NDIM
        Z=H*DERY(I)
        AUX(7,I)=Z
103     Y(I)=AUX(N,I)+.2181004*AUX(5,I)-3.[illegible]965*AUX(6,I)+3.832865*Z
C
        Z=X+H
        CALL FCT(Z,Y,DERY)
        DO 104 I=1,NDIM
104     Y(I)=AUX(N,I)+.1747603*AUX(5,I)-.5514[illegible]*AUX(6,I)
       $+1.205536*AUX(7,I)+.1711848*H*DERY(I)
        GO TO (9,13,15,21),ISW
200     ISTEP=3
201     IF(N-8) 204,202,204
202     DO 203 N=2,7
        DO 203 I=1,NDIM
        AUX(N-1,I)=AUX(N,I)
203     AUX(N+6,I)=AUX(N+7,I)
        N=7
204     N=N+1
        DO 205 I=1,NDIM
        AUX(N-1,I)=Y(I)
205     AUX(N+6,I)=DERY(I)
        X=X+H
206     ISTEP=ISTEP+1
        DO 207 I=1,NDIM
        DELT=AUX(N-4,I)+1.333333*H*(AUX(N+6,I)+AUX(N+6,I)-AUX(N+5,I)+
       $AUX(N+4,I)+AUX(N+4,I))
        Y(I)=DELT-.9256198*AUX(16,I)
207     AUX(16,I)=DELT
        CALL FCT(X,Y,DERY)
        DO 208 I=1,NDIM
        DELT=.125*(9.*AUX(N-1,I)-AUX(N-3,I)+3.*H*(DERY(I)+AUX(N+6,I)+
       $AUX(N+6,I)-AUX(N+5,I)))
        AUX(16,I)=AUX(16,I)-DELT
208     Y(I)=DELT+.07438017*AUX(16,I)
        DELT=0.
        DO 209 I=1,NDIM
209     DELT=DELT+AUX(15,I)*ABS(AUX(16,I))
```

```
      IF(DELT-EMAX   ) 210,222,222
210   CALL FCT(X,Y,DERY)
      IC=1
      H(2)=H
      H(1)=IHLF
      RETURN
211   IF(IHLF-11) 215,212,212
212   H(1)=IHLF
      RETURN
215   IF(DELT-.02*EMAX   ) 216,215,201
216   IF(IHLF) 201,201,217
217   IF(N-7) 201,218,218
218   IF(ISTEP-4) 201,219,219
219   IMOD=ISTEP/2
      IF(ISTEP-IMOD-IMOD) 201,220,201
220   H=H+H
      IHLF=IHLF-1
      ISTEP=0
      DO 221 I=1,NDIM
      AUX(N-1,I)=AUX(N-2,I)
      AUX(N-2,I)=AUX(N-4,I)
      AUX(N-3,I)=AUX(N-6,I)
      AUX(N+6,I)=AUX(N+5,I)
      AUX(N+5,I)=AUX(N+3,I)
      AUX(N+4,I)=AUX(N+1,I)
      DELT=AUX(N+5,I)+AUX(N+5,I)
      DELT=DELT+DELT+DELT
221   AUX(16,I)=8.962963*(Y(I)-AUX(N-3,I))-3.361111 *H*(DERY(I)+DELT
     1+AUX(N+4,I))
      GO TO 201
222   IHLF=IHLF+1
      IF(IHLF-10) 223,223,210
223   H=.5*H
      ISTEP=0
      DO 224 I=1,NDIM
      Y(I)=.0039625*(80.*AUX(N-1,I)+135.*AUX(N-2,I)+40.*AUX(N-3,I)+
     $AUX(N-4,I))-.1171875*(AUX(N+6,I)-6.*AUX(N+5,I)-AUX(N+4,I))*H
      AUX(N-4,I) =.00390625*(12.*AUX(N-1,I)+135.*AUX(N-2,I)+
     $108.*AUX(N-3,I)+AUX(N-4,I))-.0234375*(AUX(N+6,I)+18.*AUX(N+5,I)-
     1 9.*AUX(N+4,I))*H
      AUX(N-3,I)=AUX(N-2,I)
224   AUX(N+4,I)=AUX(N+5,I)
      X=X-H
      DELT=X-(H+H)
      CALL FCT(DELT,Y,DERY)
      DO 225 I=1,NDIM
      AUX(N-2,I)=Y(I)
      AUX(N+5,I)=DERY(I)
225   Y(I)=AUX(N-1,I)
      DELT=DELT-(H+H)
      CALL FCT(DELT,Y,DERY)
      DO 226 I=1,NDIM
      DELT=AUX(N+5,I)+AUX(N+4,I)
      DELT=DELT+DELT+DELT
      AUX(16,I)=8.962963*(AUX(N-1,I)-Y(I))-3.361111*H*(AUX(N+6,I)+DELT
```

```
      SUBROUTINE TABLE (N,ANSWER,X,XT,NX,NPX,Y,YT,NY,NPY,Z,ZT,NZ,NPZ)
C
C     TABLE LOOK-UP ROUTINE FOR 1,2 OR 3 INDEPENDENT VARIABLES
C     N      = NUMBER OF INDEPENDENT VARIABLES (ORDER)
C     N      = 1  FIRST ORDER (X)
C     N      = 2  SECOND ORDER (X,Y)
C     N      = 3  THIRD ORDER (X,Y,Z)
C     ANSWER=          (DEPENDENT VARIABLE CORRESPONDING TO INPUTS  X,Y,Z)
C     WT     = TABLE OF DEPENDENT VARIBALE CORRESPONDING TO XT,YT,ZT
C              WT(I,J,K)  INCREMENT SUBSCRIPTS LEFT TO RIGHT WHEN LOADING
C     X      = THE ARGUMENT OR INDEPENDENT VARIABLE  X
C     XT     = TABLE OF INDEP.  X  VALUES (MUST BE IN INCREASING ORDER)
C     NX     = NUMBER OF POINTS IN  XT
C     NPX    = NUMBER OF POINTS TO USE FOR  X  INTERPOLATION
C     Y      = THE ARGUMENT OR INDEPENDENT VARIABLE  Y
C     YT     = TABLE OF INDEP.  Y  VALUES (MUST BE IN INCREASING ORDER)
C     NY     = NUMBER OF POINTS IN  YT
C     NPY    = NUMBER OF POINTS TO USE FOR  Y  INTERPOLATION
C     Z      = THE ARGUMENT OR INDEPENDENT VARIABLE  Z
C     ZT     = TABLE OF INDEP.  Z  VALUES (MUST BE IN INCREASING ORDER)
C     NZ     = NUMBER OF POINTS IN  ZT
C     NPZ    = NUMBER OF POINTS TO USE FOR  Z  INTERPOLATION
C
C     REMARK  1.  THIS SUBROUTINE WILL ACCEPT  1ST, 2ND, OR 3RD  ORDER
C     REMARK  2.  IF 1ST ORDER, USE XT AND WT.
C     REMARK  3.  IF 2ND ORDER, USE XT,YT AND WT.
C     REMARK  4.  IF 3RD ORDER, USE XT,YT,ZT AND WT.
C     REMARK  5.  ALWAYS USE WT( , , ) FOR THE TABLE OF DEPENDENT VALUES.
C
      DIMENSION XT(NX),YT(NY),ZT(NZ),WT(NX,NY,NZ),W(15),A(10)
      GO TO (51,52,53), N
53    CALL LIMIT (Z,ZT,NZ,NPZ,MINZ,MAXZ)
52    CALL LIMIT (Y,YT,NY,NPY,MINY,MAXY)
51    CALL LIMIT (X,XT,NX,NPX,MINX,MAXX)
      I=1
      J=1
      GO TO (43,44,45), N
45    DO 41 J=MINZ,MAXZ
44    DO 42 I=MINY,MAXY
43    CALL INTERP (2,X,XT,WT(1,I,J),NX,NPX,MINX,MAXX,W(I))
      ANSWER=W(I)
      IF(N.EQ.1) GO TO 46
42    CONTINUE
      CALL INTERP (2,Y,YT,W,NY,NPY,MINY,MAXY,A(J))
      ANSWER=A(J)
      IF(N.EQ.2) GO TO 46
41    CONTINUE
      CALL INTERP (2,Z,ZT,A,NZ,NPZ,MINZ,MAXZ,ANSWER)
46    RETURN
      END
```

```
      SUBROUTINE LIMIT (X,XT,NX,NP ,MINX,MAXX)
C     THIS SUBROUTINE WILL FIND THE MINIMUM AND MAXIMUM SUBSCRIPTS
C     (OR RANGE) TO BE CONSIDERED FOR INTERPOLATION.
      DIMENSION XT(NX)
      NPX=NP
      IF(NPX.GT.NX) NPX=NX
      DO 25 I=1,NX
      IF(XT(I)-X) 25,22,21
25    CONTINUE
C     ..................................   GREATER THAN MAX SUBSCRIPT
24    MAXX=NX
      MINX=NX-NPX+1
      RETURN
C     ..................................   WITHIN RANGE
21    MINX=I-NPX/2
      MAXX=MINX+NPX-1
      IF(MAXX.GT.NX) GO TO 24
      IF(MINX.GE.1) GO TO 26
C     ..................................   LESS THAN MIN SUBSCRIPT
      MINX=1
      MAXX=NPX
26    RETURN
C     ..................................   NO INTERP NECESSARY
22    MINX=I
      MAXX=I
      RETURN
      END


      SUBROUTINE INTERP (LMT,X,XT,YT,NX,NPX,MINX,MAXX,Y)
C     THIS SUBROUTINE PERFORMS A SIMPLE INTERPOLATION.
C
C     INPUTS
C     LMT = 1 PROGRAM WILL DETERMINE SUBSCRIPT RANGE (MINX TO MAXX).
C     LMT = 2 PROGRAM ASSUMS THAT MINX AND MAXX SUBSCRIPTS ARE KNOWN.
C     X       ARGUMENT OR INDEPENDENT VARIABLE FOR WHICH ANSWER (Y) WILL
C             BE DETERMINED.
C     XT      TABLE OF INDEPENDENT VARIABLES.
C     YT      TABLE OF DEPENDENT VARIABLES CORRESPONDING TO XT.
C     NX      NUMBER OF POINTS IN XT,YT
C     NPX     NUMBER OF POINTS TO BE USED FOR INTERPOLATION.
C     MINX    MINIMUM XT SUBSCRIPT USED FOR INTERPOLATION.
C     MAXX    MAXIMUM XT SUBSCRIPT USED FOR INTERPOLATION.
C     OUTPUT
C     Y       ANSWER OR DEPENDENT VARIABLE CORRESPONDING TO INPUT (X).
      DIMENSION XT(NX),YT(NX)
      IF(LMT.EQ.2) GO TO 13.
      CALL LIMIT (X,XT,NX,NPX,MINX,MAXX)
130   Y=YT(MINX)
      IF(MINX.EQ.MAXX) GO TO 100
      Y=0.
      DO 120 J=MINX,MAXX
      P=1.
      DO 110 I=MINX,MAXX
      IF(I.EQ.J) GO TO 110
      P=P*(X-XT(I))/(XT(J)-XT(I))
110   CONTINUE
120   Y=Y + YT(J)*P
100   RETURN
      END
```

```
      SUBROUTINE QSDSUB
      REAL MACH,KE,K
      INTEGER SX,TX
      COMMON/XXX/VM2
      COMMON /AIR/ MACH,VM,VS,QS,QSD,QSR,RHO
      COMMON/OTHER2/ THTA,PSI,PHI,KE,S,D,SX,TX,K,G,MAXPT
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C *   COMPUTE DYNAMIC PRESSURE TERMS                                  *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      QS=RHO*VM2*S
      QSD=QS*D
      QSR=RHO*VM*S*D*D
      RETURN
      END


      SUBROUTINE PLOT(A,N,M,NL,NS)
C
C     ...................................................................
C
C     SUBROUTINE PLOT
C
C     PURPOSE
C     PLOT SEVERAL CROSS-VARIABLES VERSUS A BASE VARIABLE
C
C     USAGE
C     CALL PLOT(A,N,M,NL,NS)
C
C     DESCRIPTION OF PARAMETERS
C     A - MATRIX OF DATA TO BE PLOTTED) FIRST COLUMN REPRESENTS
C     BASE  VARIABLE AND SUCCESSIVE COLUMNS ARE THE CROSS-
C     VARIABLES (MAXIMUM IS 9).
C     N - NUMBER OF ROWS IN MATRIX A
C     M - NUMBER OF COLUMNS IN MATRIX A (EQUAL TO THE TOTAL
C     NUMBER OF VARIABLES). MAXIMUM IS 10.
C     NL - NUMBER OF LINES IN THE PLOT. IF 0 IS SPECIFIED, 50
C     LINES ARE USED.
C     NS - CODE FOR SORTING THE BASE VARIABLE DATA IN ASCENDING
C     ORDER
C     0 SORTING IS NOT NECESSARY (ALREADY IN ASCENDING
C     ORDER).
C    1 SORTING IS NECESSARY.
C
C     REMARKS
C     NONE
C
C     SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C     NONE
C
C     ...................................................................
C
      DIMENSION OUT(101),YPR(11),ANS(9),A(1)
      DATA BLANK,ANG(1),ANG(2),ANG(3),ANG(4),ANG(5),ANG(6),ANG(7),ANG(8)
    2,ANG(9)/1H ,1H*,1H+,1H.,1HX,1HO,1HO,1HA,1HB,1HC/
C
    2 FORMAT(1H ,F11.4,5X,101A1)
    3 FORMAT(1H )
    4 FORMAT(10H 123456789)
    5 FORMAT(16/
  7   FORMAT(//30X,*HORIZONTAL AXIS RESOLUTION + OR - ***F11.4/
     X30X,*VERTICAL AXIS RESOLUTION +OR- ***F11.4)
  8   FORMAT(1H0,11X,F12.5,84X,F12.5)
C     ...................................................................
C
C
      NLL=NL
C
      IF(NS) 16, 16, 10
C
C     SORT BASE VARIABLE DATA IN ASCENDING ORDER
C
```

```
   10 DO 15 I=1,N
      DO 14 J=1,N
      IF(A(I)-A(J)) 14, 14, 11
   11 L=I-N
      LL=J-N
      DO 12 K=1,M
      L=L+N
      LL=LL+N
      F=A(L)
      A(L)=A(LL)
   12 A(LL)=F
   14 CONTINUE
   15 CONTINUE
C
C     TEST NLL
C
   16 IF(NLL) 20, 18, 20
   18 NLL=50
C
C
20    CONTINUE
C
C     FIND SCALE FOR BASE VARIABLE
C
      XSCAL=(A(N)-A(1))/(FLOAT(NL -1))
C
C     FIND SCALE FOR CROSS-VARIABLES
C
C
      M1=N+1
      YMIN=A(M1)
      YMAX=YMIN
      M2=M*N
      DO 40 J=M1,M2
      IF(A(J)-YMIN) 28,26,26
   26 IF(A(J)-YMAX) 40,40,30
   28 YMIN=A(J)
      GO TO 40
   30 YMAX=A(J)
   40 CONTINUE
      YSCAL=(YMAX-YMIN)/100.0
C
C     FIND BASE VARIABLE PRINT POSITION
C
      XB=A(1)
      L=1
      MY=M-1
      I=1
   45 F=I-1
      XPR=XB+F*XSCAL
      IF(A(L)-XPR) 50,50,70
C
C     FIND CROSS-VARIABLES
C
   50 DO 55 IX=1,101
```

```
   55 ~~OUT(IX)=BLANK~~
      IF(I.EQ.1.OR.I.EQ.NLL)  GO TO 101
      OUT(1)=OUT(101)=ANG(3)
      GO TO 102
 101  DO 100 I=1,101,10
 100  OUT(I)=ANG(3)
 102  CONTINUE
      DO 60 J=1,MY
      LL=L+J*N
      JP=((A(LL)-YMIN)/YSCAL)+1.0
      OUT(JP)=ANG(J)
   60 CONTINUE
      IF(I.NE.NLL)  GO TO 200
      IF(YMIN.GE.0.)  GO TO 200
      IF(YMAX.LE.0.)  GO TO 200
      IZERO=-YMIN/YSCAL+1.
      OUT(IZERO)=ANG(5)
 200  CONTINUE
C
C     PRINT LINE AND CLEAR, OR SKIP
C
      WRITE(6,2)XPR,(OUT(IZ),IZ=1,101)
      L=L+1
      GO TO 80
   70 WRITE(6,3)
   80 I=I+1
      IF(I-NLL) 45,84,86
   84 XPR=A(N)
      GO TO 50
C
C     PRINT CROSS-VARIABLES NUMBERS
C
 86   CONTINUE
      YPR(1)=YMIN
      DO 90 KN=1,9
   90 YPR(KN+1)=YPR(KN)+YSCAL*10.0
      YPR(11)=YMAX
      WRITE(6,8) YPR(1),YPR(11)
      WRITE(6,7)  YSCAL,XSCAL
      RETURN
      END
```

```
SUBROUTINE LAG(INPUT,OUTPUT,X,DX,INDEX,TIMCON)
REAL INPUT
DIMENSION X(1),DX(1)
DX(INDEX) =(-X(INDEX)+INPUT)/TIMCON
OUTPUT=X(INDEX)
RETURN
END


SUBROUTINE SECORD(INPUT,OUTPUT,X,DX,INDEX,ETA,WN)
DIMENSION X(1),DX(1)
REAL INPUT
W2=WN*WN
DX(INDEX)=X(INDEX+1)
DX(INDEX+1)=-2.*ETA*WN*X(INDEX+1)-W2*X(INDEX)+W2*INPUT
OUTPUT=X(INDEX)
RETURN
END


SUBROUTINE LIM(INPUT,OUTPUT,XMIN,XMAX)
REAL INPUT
IF(INPUT.GT.XMAX)  OUTPUT=XMAX
IF(INPUT.LT.XMIN) OUTPUT=XMIN
RETURN
END


SUBROUTINE LIMSTA(INPUT,OUTPUT,XMIN,XMAX)
REAL INPUT
IF(INPUT.GT.XMAX) OUTPUT=INPUT=XMAX
IF(INPUT.LT.XMIN) OUTPUT=INPUT=XMIN
RETURN
END


      SUBROUTINE DEADSP(INPUT,OUTPUT,LOWER,UPPER)
      REAL INPUT
      REAL LOWER
      IF(INPUT.GT.LOWER.AND.INPUT.LT.UPPER)  GO TO 1
      IF(INPUT.LE.LOWER)  GO TO 2
      OUTPUT=INPUT-UPPER
      RETURN
2     OUTPUT=INPUT-LOWER
      RETURN
1     OUTPUT=0.
      RETURN
      END
```

```
      SUBROUTINE DETECT(INPUT,OUTPUT,SLOPE,MAXLRO,MAXNLO,FOV)
      REAL INPUT
      REAL MAXLRO,MAXNLO
      IF(ABS(INPUT).GT.FOV)  GO TO 1
      OUTPUT=INPUT*SLOPE
      IF(ABS(OUTPUT).GT.MAXLRO)  OUTPUT=SIGN(MAXNLO,OUTPUT)
      RETURN
1     OUTPUT=0.
      RETURN
      END
```

```
      SUBROUTINE GYRO2(TGY,TGZ,X,DX,INDEX,IX,ITP,ITY,WS)
      REAL LY,LZ
      REAL IX,ITP,ITY
      DIMENSION X(1),DX(1)
      THTAG=X(INDEX)
      PSIG=X(INDEX+1)
      CPG=COS(PSIG)
      SPG=SIN(PSIG)
      CTG=COS(THTAG)
      STG=SIN(THTAG)
      P=X(1)
      Q=X(2)
      RR=X(3)
      LY=TGY*CPG+TGZ*STG*SPG
      LZ=TGZ*CTG
      PG=(P*CTG-RR*STG)/CPG
      QG=-LZ/(PG*(IX-ITP)+WS*IX)
      RG=LY/(PG*(IX-ITY)+WS*IX)
      DX(INDEX)=QG/CPG+PG*SPG-Q
      DX(INDEX+1)=RG-RR*CTG-P*STG
      RETURN
      END
```

```
      SUBROUTINE LDLAG(INPUT,OUTPUT,X,DX,INDEX,TMCON1,TMCON2)
      REAL INPUT
      DIMENSION X(1),DX(1)
      OUTPUT=(X(INDEX)+TMCON1*INPUT)/TMCON2
      DX(INDEX)=INPUT-OUTPUT
      RETURN
      END
```